Proyecto de Sistemas Informáticos. Madrid Live: un sistema de recomendación de ocio social y contextual para la ciudad de Madrid

MEMORIA DEL PROYECTO

Javier López Gómez
Marina Bezares Álvarez
Roberto Marín Fernández

Dirigido por: Belén Díaz Agudo
Codirigido por: Juan A. Recio García

Departamento de Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid

Junio 2014

Proyecto de Sistemas Informáticos. Madrid Live: un sistema de recomendación de ocio social y contextual para la ciudad de Madrid

Memoria de Proyecto de Sistemas Informáticos
Ingeniería del Software e Inteligencia Artificial
06/2014

Departamento de Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid

Junio 2014

Autorización

Javier López Gómez, Marina Bezares Álvarez y Roberto Marín Fernández, alumnos matriculados en la asignatura de Sistemas Informáticos, autorizan, mediante el presente documento, a la Universidad Complutense de Madrid (UCM), a difundir y utilizar con fines académicos, no comerciales, y mencionando expresamente a sus autores, tanto la propia memoria, como el código, la documentación y/o el prototipo desarrollado. Todo ello realizado durante el curso académico 2013-2014 bajo la dirección de Belén Díaz Agudo y Juan Antonio Recio, profesores del Departamento de Ingeniería del Software e Inteligencia Artificial.

Javier López Gómez Marina Bezares Álvarez Roberto Marín Fernández

Agradecimientos

Aunque estas líneas a menudo son pasadas por alto, para nosotros son de gran importancia y queremos aprovecharlas para dar los tan bien merecidos agradecimientos a todas aquellas personas que nos han permitido llegar a donde estamos ahora mismo.

El primero de estos agradecimientos va para Belén Díaz Agudo y Juan Antonio Recio, bajo cuya dirección se ha llevado a cabo este proyecto. Gracias tanto por haber confiado en nosotros para la continuación de un proyecto tan ilusionante como ha sido el de Madrid Live como por todo el tiempo dedicado a reuniones, resolución de dudas y presentaciones, que tan útil nos ha sido para llevar este proyecto a buen puerto.

Gracias también a todos los compañeros que han estado ahí durante todos estos años de carrera ofreciendo su ayuda siempre que era necesaria. Es un orgullo haber podido trabajar con todos vosotros. Queremos hacer, eso sí, una mención especial a todos aquellos que nos habéis echado una mano con el experimento de validación de Madrid Live.

No olvidamos tampoco el agradecimiento bien merecido por todos los profesores que hemos tenido a lo largo de la carrera, ya que han sido las personas que nos han moldeado y atendido durante tantos años para conseguir hacer de nosotros los buenos Ingenieros que somos y seremos. Agradecer también su labor a todo el personal adicional de la facultad (secretaría, conserjería, etc.), siempre atento y tan trabajador.

Por último, y siendo uno de los más importantes, un agradecimiento especial a nuestras familias, ya que gracias a su apoyo incondicional y ayuda hemos podido terminar esta carrera y convertirnos en los profesionales que durante tantos años hemos querido ser.

A todos vosotros, que habéis mostrado vuestro apoyo durante varios años, os dedicamos este proyecto de fin de carrera.

Resumen

En la actualidad muchos servicios web disponen de un sistema recomendador que orienta al usuario en la búsqueda de un producto determinado como pueden ser Amazon o TripAdvisor. Los sistemas de recomendación se han utilizado ampliamente tanto en la venta de productos (pisos, libros, ordenadores...) como en el sector turístico (viajes, hoteles...).

En este proyecto hemos desarrollado Madrid Live, un sistema de recomendación de ocio social y contextual para la ciudad de Madrid cuya interfaz es una aplicación móvil. Se trata de un sistema social ya que está orientado a grupos y contextual porque recoge y procesa toda la información posible del contexto que rodea a las recomendaciones.

El principal objetivo de Madrid Live es la generación de planes de ocio que incluyan distintos tipos de actividades de forma que se ajusten de la mejor forma posible tanto a los parámetros de la solicitud generada por el usuario como a los gustos y preferencias del mismo. Para el prototipo hemos incluido museos, restaurantes, parques, paseos y cines pero el sistema es fácilmente extensible a otros tipos de actividades. Además, el sistema es capaz de usar toda la información contextual recuperada para reducir al mínimo la información que proporciona explícitamente el usuario a la hora de solicitar un plan de ocio.

Para recuperar la información de contexto se han aprovechado todos los servicios ofrecidos por la tecnología Android además de implementar otros módulos propios como un sistema de geolocalización que permite localizar amigos cercanos. En lo que a la parte social y orientada a grupos se refiere, se ha integrado Madrid Live con un servicio de red social (Facebook en la implementación actual) del que se recupera toda la información posible acerca de los usuarios para conseguir una visión detallada de su círculo social y las relaciones existentes con los diferentes amigos.

Palabras clave: sistemas de recomendación, aplicación móvil, datos de contexto, datos sociales, CBR.

Abstract

Nowadays lots of web services make use of recommender systems to lead the user while he searches for a desired product as, for example, Amazon[1] or TripAdvisor[2]. Recommender systems have been highly used in products sales (books, computers, flats...) as well as in the travel and tourism sector (travels, hotels...).

In this project we've developed Madrid Live, which is a social and contextual leisure recommender for the city of Madrid and which has a mobile application as interface. It's a social system because it's very directed to groups of users and a contextual one because it collects and uses all available information of the context of every recommendation.

The main objective of Madrid Live is to create leisure plans which can include different types of activities. In the current prototype we've included museums, restaurants, parks, walks and cinemas but the system can be easily extended to include more activities. Every plan will fit the users' tastes and preferences in the best possible way. In addition, the system is able to use all the retrieved contextual information to reduce the number of explicit information introduced by the user to request a leisure plan.

In order to collect all the contextual data, all available services offered by the Android technology have been used and another modules have been implemented (as a custom geolocalization system to get the user's nearby friends). Regarding to the social and group-oriented side, Madrid Live has been integrated with a social network service (Facebook in the current implementation) from which all available users' data is retrieved in order to get a more detailed view about their social circle and their relations with other friends.

Keywords: recommender systems, mobile application, context data, social data, CBR.

[1] http://www.amazon.es/
[2] http://www.tripadvisor.es/

Índice

Índice de figuras

Índice de Tablas

Capítulo 1

Introducción

En este capítulo explicaremos qué ha motivado la realización de este proyecto y el desarrollo de Madrid Live, cuales son los objetivos que se pretenden conseguir con el proyecto y mostraremos una breve descripción de cómo está estructurada la memoria.

1.1. Motivación

La ciudad de Madrid ofrece un gran número de actividades de ocio para sus habitantes y visitantes. Cuando una persona va a planificar una serie de actividades, típicamente busca información por Internet de qué puntos de interés visitar, qué restaurantes son los más recomendados para ir a comer o a qué hora ponen una película en un determinado cine.

Para esta búsqueda el usuario puede hacer uso de un recomendador de actividades de ocio como, por ejemplo, TripAdvisor[1], que es un ejemplo de entre muchos tipos de páginas web en las que el usuario introduce sus preferencias y el sistema muestra una lista de actividades para que elija la que más se adecue a sus gustos. Madrid Live se basa en estos sistemas de recomendación para facilitar la búsqueda de actividades de ocio en la ciudad de Madrid.

Dentro de los sistemas de recomendación no solo hay recomendadores individuales sino que también existen recomendadores grupales, que tienen en cuenta las preferencias de cada miembro y recomiendan actividades que satisfagan lo mejor posible a todos los componentes grupo. Cuando una persona va a realizar una serie de actividades por Madrid es muy común que vaya acompañado por otras personas, por lo que Madrid Live incorpora un sistema de recomendación grupal en el que se tienen en cuenta las relaciones sociales de las personas que componen el grupo, de forma que se facilita y

[1] http://www.tripadvisor.es/

mejora la búsqueda de actividades.

Los sistemas de recomendación pueden obtener información del contexto para lograr una mayor precisión en la recomendación. Esta información puede ser dinámica – como la localización de la persona o el tiempo que hace en un determinado momento – o estática – como, por ejemplo, los gustos del usuario –. Generalmente es el usuario quien introduce toda esta información, lo que puede resultar tedioso para él, pero nuestro sistema facilita y automatiza la recogida de todos estos datos aprovechando al máximo las capacidades de las tecnologías móviles.

La motivación de este trabajo es permitir al usuario planificar actividades de ocio por la ciudad de Madrid ofreciendo siempre aquellas que mejor encajen con sus gustos. En el caso de las recomendaciones para grupos, el objetivo es tener en cuenta las relaciones sociales entre todos los componentes para recomendar actividades, las cuales gusten a todas las personas que integran el grupo. Todo ello con la pulsación de un solo botón y rellenando la mínima información, facilitando así a los usuarios poder conocer todos los rincones turísticos de ofrecidos por la ciudad de Madrid.

1.2. Objetivos

El principal objetivo de Madrid Live es ser una aplicación móvil cómoda de utilizar y útil para organizar visitas, sobre todo en grupo, por la ciudad de Madrid. Se pretende que en estas recomendaciones para grupos todos sus componentes queden satisfechos con las actividades propuestas, para lo que se recoge el contexto social de cada uno de los miembros obteniendo estos datos de la red social elegida. También se ha hecho especial hincapié en mejorar el modelado del perfil de preferencias del usuario así como automatizar lo máximo posible la recogida de datos para la recomendación.

Para todo esto, en Madrid Live se presentan los siguientes objetivos:

1. Migración del proyecto original a una plataforma móvil que permita recopilar información de contexto.

2. Crear un sistema de geolocalización propio que permita obtener información acerca de la posición del usuario y los amigos cercanos.

3. Realizar una integración con algún servicio externo o red social para la recuperación de información del círculo social de los usuarios.

4. Recopilar toda la información de contexto posible con el fin de automatizar y mejorar el sistema de recomendación minimizando el número de entradas manuales del usuario.

5. Mejorar los datos y modelado del perfil de preferencias del usuario.

6. Llevar a cabo mejoras sobre el sistema de recomendación grupal.

7. Incluir nuevas actividades de ocio al margen de las originales (museos, paseos, parques y restaurantes).

1.3. Estructura de la memoria

Esta memoria está organizada de la siguiente manera:

1. En este primer capítulo se han expuesto la motivación y los objetivos que se pretenden cumplir con Madrid Live y se explica brevemente cómo va a estar estructurada la memoria.

2. El Capítulo 2 presenta el estado del arte. En esta parte se describirán brevemente los sistemas de recomendación actuales tanto individuales como grupales. Después, se definirá el concepto de contexto y se explicará su relación con los recomendadores y, por último, se explicarán las tecnologías que se han utilizado para la implementación de Madrid Live.

3. A continuación, en el Capítulo 3 se detallarán todas las fuentes de conocimiento del sistema, la información de contexto recogida y los datos de modelado del perfil de preferencias del usuario. También se ofrecerá una amplia visión de las diferentes funcionalidades de Madrid Live así como de la interfaz Android.

4. El Capítulo 4 servirá de complemento al anterior ya que en él se explicarán de forma mucho más detallada tanto la arquitectura como el funcionamiento de cada uno de los módulos que componen Madrid Live. Para ello se empezará dando una visión global de la arquitectura del sistema para pasar a hablar de las funcionalidades generales, de la capa de servicios y del cliente Android.

5. En el Capítulo 5 mostraremos los resultados de un experimento de validación llevado a cabo para Madrid Live, así como los resultados y conclusiones obtenidas del mismo.

6. El Capítulo 6 expondrá los objetivos cumplidos en Madrid Live además de las conclusiones generales sacadas de todo el proyecto.

7. Como complemento a los capítulos 5 y 6, el Capítulo 7 mostrará varias líneas de investigación y desarrollo futuras que pueden ser llevadas a cabo a partir del estado actual del proyecto.

8. Por último, en los apéndices se incluye información complementaria acerca del sistema. En concreto: se muestran los servicios que proporciona la API, las tablas que componen la base de datos y los casos de uso y diagramas de flujo de las diferentes funcionalidades del sistema.

Capítulo 2

Estado del arte

2.1. Sistemas de recomendación

Estamos en la era de la información. En nuestro día a día se nos bombardea continuamente con nueva información de todo tipo. Este exceso de información puede ser muy abrumador, hasta el punto de que no sepamos escoger lo que queremos o lo más adecuado para nosotros entre las alternativas que se nos ofrecen. Los *sistemas de recomendación* (Bridge et al., 2006a) surgen como solución a este problema, ya que su principal función es procesar la información disponible y devolver un número reducido de opciones entre las que se espera encontrar la más adecuada para el usuario que ha solicitado la recomendación.

2.1.1. Recomendadores individuales

El objetivo de los sistemas recomendadores individuales es ayudar a los usuarios en la toma de decisiones presentando aquellos elementos que más le puedan interesar. Aunque esta tecnología tiene muchas áreas de aplicación debemos destacar el uso comercial que se ha hecho de ella. Por ejemplo, empresas como la Casa del Libro[1] o Amazon[2] utilizan los sistemas de recomendación para presentar a los usuarios los productos que se espera estén más dispuestos a comprar.

Según la metodología empleada por los recomendadores podemos dividir los sistemas de recomendación en dos tipos:

1. **Basados en contenido.** El recomendador se focaliza en buscar productos cuyas características se ajusten los mejor posible a las preferencias del usuario. Es decir, se disecciona la información de las par-

[1] www.lacasadellibro.com
[2] www.amazon.es

ticularidades, propiedades y cualidades de los productos en base a su descripción y se recomienda al usuario aquel cuyas características se ajusten mejor a lo que busca. El razonamiento de este tipo de recomendador es que si a un usuario le gustan productos de una determinado ámbito, también le gustarán otros productos de características similares a ellos.

2. **Filtrado colaborativo.** El recomendador se centra en buscar productos que otros usuarios con gustos similares hayan valorado positivamente. Es decir, los usuarios expresan su opinión sobre productos ya conocidos y se recomiendan los productos que tengan el mejor prestigio entre la comunidad de usuarios. El razonamiento de este tipo de recomendador es que si algo le ha gustado a gran cantidad de personas similares al usuario, lo más seguro es que a él también le guste.

En Madrid Live hemos elegido implementar un sistema de recomendación *basado en contenido*. En la Sección 3.3 se detallará cómo recopilamos la información sobre los gustos y preferencias del usuario.

2.1.2. Sistemas de recomendación de ocio en Madrid

Antes de proseguir detallando en más profundidad los sistemas de recomendación, vamos a hacer un análisis de uno de ellos: Fever. Hemos escogido para el análisis este recomendador porque, al igual que Madrid Live es un sistema de recomendación de ocio en Madrid, es decir, porque trata el mismo ámbito que nuestro recomendador.

Fever recomienda una serie de actividades de ocio en Madrid para el usuario basándose en sus preferencias, obtenidas a partir de su cuenta Facebook, con la que es necesario identificarse, y de un test de preferencias realizado tras el registro, lo que indica que se trata de un recomendador basado en contenido, ya que se busca actividades que encajen lo mejor posible con las que las preferencias del usuario y condiciones de los eventos.

Las actividades que Fever oferta son eventos patrocinados por empresas y particulares que se quieren anunciar. Estos eventos tienen fecha y se catalogan por actividades para hacer hoy, mañana o más tarde. Si un evento te interesa puedes apuntarte cómo que vas a asistir. Algunos eventos son de pago y se puede comprar la entrada desde la propia aplicación.

Fever también tiene una parte social en la aplicación ya que un usuario puede seguir la actividad de otros y la aplicación le informará de a qué actividades se han apuntado esas personas. A pesar de ello, no podemos decir que Fever sea un recomendador colaborativo en sentido estricto ya que no

recomienda esas actividades directamente, aunque espera que el usuario se fije en ellas y que se anime a participar.

En conclusión, Fever es un sistema que compara actividades basándose principalmente en la información obtenida de las redes sociales y las preferencias introducidas por el usuario de manera explícita, aunque también intenta introducir un aspecto social en la recomendación permitiendo seguir la agenda de actividades de otros usuarios. Está orientado empresarialmente puesto que solo recomienda actividades patrocinadas en la que diferentes empresas publicitan sus eventos. Comparte con Madrid Live la idea de la recomendación de actividades de ocio en la ciudad de Madrid y la recolección de datos de un servicio externo (Facebook).

2.1.3. Recomendadores grupales

Los recomendadores tradicionales recomiendan un producto, o una serie de productos a un usuario. ¿Pero qué ocurre si tiene que recomendar a varios usuarios el mismo producto? Nos referimos a grupos de usuarios que van a consumir el mismo producto recomendado. De esta necesidad nacen los *recomendadores grupales* (Masthoff, 2011) , que recomiendan un producto a varios usuarios a la vez. En particular, esta situación se da en los recomendadores de planes de ocio, como Madrid Live, en los que se recomienda un plan que, se espera, todos los los miembros un grupo realizarán.

La principal dificultad de los recomendadores grupales reside en cómo casar las inclinaciones y preferencias individuales de cada uno de los miembros del grupo de manera que la recomendación resultante satisfaga lo máximo posible a todos por igual.

Hay diversas soluciones que permiten combinar las distintas recomendaciones individuales generadas para componer una recomendación grupal, como por ejemplo:

1. **Mezclar las recomendaciones individuales.** Se escogen las recomendaciones individuales y se presenta una variedad de las diferentes recomendaciones individuales.

2. **Crear una agregación.** Se valoran las recomendaciones individuales puntuando de manera más alta cuanto mejor se estime que es esa recomendación para el individuo. Luego, para cada individuo se escoge el conjunto de recomendaciones que tenga la mayor puntuación. Mediante una fórmula se agregan todas las recomendaciones del conjunto final y se determina cuál es la mejor.

3. **Tratar de manera diferente** a cada miembro del grupo. Permite

asignar distintos pesos a los miembros del grupo en función de su importancia dentro del mismo.

4. **Crear un modelo del grupo** como si fuera un individuo y hacer una recomendación individual tradicional sobre dicho modelo.

De las distintas posibilidades, para Madrid Live hemos escogido la opción de recomendación basada en funciones de agregación, que son fórmulas que trabajan con las puntuaciones de las recomendaciones individuales y determinan qué actividad es la que mejor encaja con los gustos generales del grupo. Tras barajar las diferentes funciones de agregación clásicas, decidimos hacer uso de la *agregación media*.

$$R_i = media(r_{ij}) = \frac{1}{n} \sum_{j=1}^{n} r_{ij}$$

Donde r_{ij} es la valoración de cada usuario u_i para un producto p_j y R_i el rating final que obtiene el producto p_j para el grupo. Cada usuario u_i valora cada producto pj, dando lugar a r_{ij}. Luego, se calcula puntuación media de cada producto, R_i, y se escoge el producto que haya obtenido la mayor valoración media.

La función de agregación media puede ser bastante plana, sin embargo, la hemos escogido por una razón, y es que es la función de agregación básica considerada más justa y equilibrada. En Madrid Live hemos utilizado la novedosa aproximación de Lara Quijano Sánchez (2010) consistente en incluir factores sociales en la función de agregación. Concretamente, el *trust* o valor de confianza, que se aplica como un peso que pondera y enriquece la función de agregación media (se detallará en la Sección 3.4.7).

2.1.3.1. Trust o valor de confianza

Un grupo que vaya a pasar una tarde de ocio junto puede ser muy variado, amigos de toda la vida, amigos que no se vean muy a menudo, una pareja, una familia con hijos etc. Cuando somos miembros de un grupo, no tenemos el mismo grado de amistad con todos los miembros del grupo y por tanto, todas las opiniones de los diferentes miembros del grupo no nos importan de la misma manera. Esta dinámica es la que hemos querido reflejar mediante el *trust o valor de confianza* (Golbeck, 2006a) (Golbeck, 2006b).

El *trust* o confianza es un valor que pretende medir la confianza o intimidad que tienen dos personas entre sí. El cálculo se hace partiendo de un modelo propuesto en otro proyecto (Lara Quijano Sánchez, 2010) y cuyo valor final vendrá determinado por 4 parámetros adicionales denominados: *intimacy*, *intensity*, *duration* y *reciprocal services*.

1. **Intimacy.** Intimidad. Pretende medir la intimidad o cercanía de la relación.
2. **Intensity.** Intensidad. Tiene como objetivo calcular la intensidad de la relación. Es decir, si la amistad tiene contacto diario o esporádico.
3. **Duration.** Duración. Mide la solidez y estabilidad de la relación en el tiempo.
4. **Reciprocal services.** Entrega mutua. Pretende estimar la entrega mutua de los componentes de la relación.

Mediante la inclusión del *trust* lo que conseguimos es que las personas con más confianza por parte del grupo tengan más peso a la hora de la elección final de la actividad.

Para realizar el cálculo del trust extraemos información del usuario a través de sus redes sociales, en concreto de Facebook, que nos permite generar un perfil social del usuario frente a otros cuando se encuentran en un grupo. Los detalles de qué información se recopila y cómo los veremos en la Sección 4.1.2.

2.2. Fuentes de información en los sistemas de recomendación

De manera intuitiva se puede vislumbrar la importancia de la recopilación de información a hora de recomendar. Es necesario conocer las características de lo que se recomienda y conocer también al usuario para poder realizar una buena recomendación.

En los sistemas recomendadores hay dos formas principales de recopilar información del usuario explícitamente e implícitamente.

- **Explícitamente.** Se pregunta directamente al usuario por la información que se quiere obtener.
- **Implícitamente.** La información se deduce o se adquiere mediante otros métodos que no sean la pregunta directa.

Si únicamente tuviésemos información de manera explícita, deberíamos hacer demasiadas preguntas al usuario y por tanto correr el riesgo de que se abrume o se aburra y no termine de responderlas. Por ello, en Madrid Live nos hemos centrado en que la mayor parte de la información sea adquirida implícitamente es decir, de forma transparente al usuario. A continuación presentaremos el tipo de información utilizada en Madrid Live.

2.2.1. Información de contexto

Con el objetivo de recomendar la mejor alternativa para el usuario entre un grupo de opciones, los recomendadores han de tener en cuenta un gran número de variables. Algunas de las cuales son altamente conocidas y usadas, como los gustos y preferencias del usuario, pero hay otros factores que pueden afectar a la recomendación que no dependen directamente de usuario sino del entorno en el que se encuentra. Para realizar una buena recomendación hay que tener en cuenta todas las variables que influyen en la elección de un producto.

El contexto (Bridge et al., 2006b) engloba toda la información sobre condiciones y circunstancias que pueden afectar a la recomendación. Es un concepto muy amplio y tiene varias facetas. Decidimos que centrarnos en una sola dimensión del contexto era insuficiente y, por ello, en Madrid Live manejamos diferentes tipos de contexto. Además, así hacemos más rica la recomendación.

Los diferentes aspectos del contexto que contemplamos en Madrid Live son:

- **Contexto personal.** Son las circunstancias personales del individuo que puedan afectar a la recomendación. Es decir, esencialmente los gustos y preferencias personales.
- **Contexto social.** Engloba principalmente al grupo cercano al usuario y se centra en el distinto grado de relación que haya entre el usuario y los otros componentes del grupo.
- **Contexto físico.** Hace referencia a todas las circunstancias físicas que puedan afectar al usuario como la hora actual, el lugar donde se encuentra, el tiempo atmosférico etc.

2.2.1.1. Contexto personal

El *contexto personal* (Gedikli, 2013) es en esencia los gustos y las preferencias del usuario. Primero, para recopilar el contexto personal del usuario, le presentamos una serie de test de preferencias que nos ayuden a conocerlo mejor. Con las respuestas del test confeccionamos el *perfil de usuario*, que es una manera de sintetizar la información del contexto personal.

Los test de preferencias son la manera que tenemos de arrancar y de conocer inicialmente el contexto personal de cada usuario. Sin embargo, los seres humanos no somos estáticos y nuestras inclinaciones pueden variar con el tiempo. Por ello, cuando presentamos al usuario una recomendación éste

puede rechazarla si no le gusta o aceptarla. Si la acepta, puede puntuar la recomendación para valorar cuánto le ha gustado. Estas valoraciones no caen en saco roto, sino que recopilamos información y con ella ponderamos, más aún, los gustos del usuario. Además, es una manera transparente de recopilar información de contexto personal, muy cómoda para el usuario.

2.2.1.2. Contexto social

El *contexto social* son todos aquellos factores que hay que tener en cuenta cuando un grupo de personas queda para un plan como cuánta gente ha quedado, quienes son, las relaciones entre ellos, qué le gusta a cada uno etc. El cálculo del trust es una parte importante del contexto social puesto que expresa las relaciones que hay entre los miembros de un grupo sin embargo no son lo mismo.

El contexto social (Gilbert, 2009) que usamos en Madrid Live está compuesto principalmente por los *amigos cercanos* y toda la información de los círculos sociales del usuario recuperada de Facebook. Haciendo uso de un mecanismo de posicionamiento y con ayuda de la sincronización con el servidor de Madrid Live, el sistema es capaz de detectar qué amigos se encuentran cerca del usuario (100 metros como mucho) y los agrega automáticamente al plan. Con el conocimiento de cuál es la composición concreta del grupo que va a realizar el plan calculamos el valor de *trust* entre cada par de miembros y lo usamos para mejorar la recomendación.

Las *redes sociales* son un lugar dónde los usuarios comparten en comunidad. Lo que comparten es muy variado como gustos, opiniones, expresiones, experiencias etc. Es un modelo social en alza que ha surgido recientemente pero que se ha consolidado de tal manera que las redes sociales son una parte muy importante de las relaciones sociales. Por esta razón, de compartir, las redes sociales es el lugar idóneo donde recopilar información sobre los individuos que componen un grupo. La información extraída de las redes sociales es vital para Madrid Live ya que es imprescindible en la obtención toda la información necesaria para el contexto social. Los detalles de qué información se recopila y cómo los veremos en la Sección 4.1.2.

2.2.1.3. Contexto físico

Seguro que a todos nos ha pasado de querer ir al parque cuando está lloviendo, o de meternos en el cine un día precioso, o de ir al museo y que esté cerrado por ser lunes, porque hayamos llegado muy tarde, o por ser festivo. Todas estas circunstancias que hemos mencionado son circunstancias físicas propias del entorno que rodea y particulares del momento en el que se hace la recomendación. En Madrid Live analizamos estas circunstancias

físicas o *contexto físico* para ofrecer una recomendación más completa. Los parámetros del contexto físico que analizamos son:

- **Fecha y hora.** Por defecto, el cliente Android de Madrid Live es capaz de reconocer la fecha y hora del día y fijará un intervalo para el plan basado en ellas. Además, establecerá los límites necesarios para garantizar que el plan se crea solamente para un día. El usuario también podrá elegir una hora de inicio y final libremente.
- **Posición.** Gracias a un sistema de geolocalización propio, la aplicación cliente tendrá siempre, con permiso del usuario, la última posición conocida del mismo. Esta información se facilitará al sistema de recomendación, lo cual permitirá que éste seleccione las actividades más cercanas posibles a dicha posición.

2.3. Razonamiento basado en casos. CBR

Una vez recopilada toda la información necesaria para realizar una buena recomendación hay que implementar el recomendador. Para ello, hemos utilizado un sistema de *razonamiento basado en casos* (CBR, del inglés Case Based Reasoning) ya que estos sistemas se prestan fácilmente a implementar recomendadores basados en contenido. En esta sección vamos a explicar brevemente qué son y cómo funcionan.

El Razonamiento basado en casos es el proceso de solucionar nuevos problemas basándose en las soluciones de problemas anteriores, es una manera de razonar haciendo analogías. Estos sistemas trabajan con una *base de casos*, los cuales son descripciones de problemas concretos que han ocurrido anteriormente y que tienen asociada la solución que se siguió para resolver ese problema. Cuando se presenta un nuevo problema puede suceder que haya sido resuelto con anterioridad, en ese caso se recupera la solución del caso y se aplica (experiencia). También puede ocurrir que el problema no haya sido resuelto anteriormente y no tengamos la solución, en ese caso lo resolvemos buscando un problema parecido que haya sucedido anteriormente y proponiendo la misma solución o una solución similar. Si la solución encontrada para el nuevo caso fue un éxito, se guarda en la base de casos y pasa a formar parte de la experiencia.
El *ciclo CBR* describe todo el proceso que realiza el CBR cuando llega un caso nuevo.

1. **Recuperar.** Extraemos los casos más similares al caso nuevo
2. **Reutilizar.** Elegimos el mejor caso entre los recuperados y consultamos su solución.

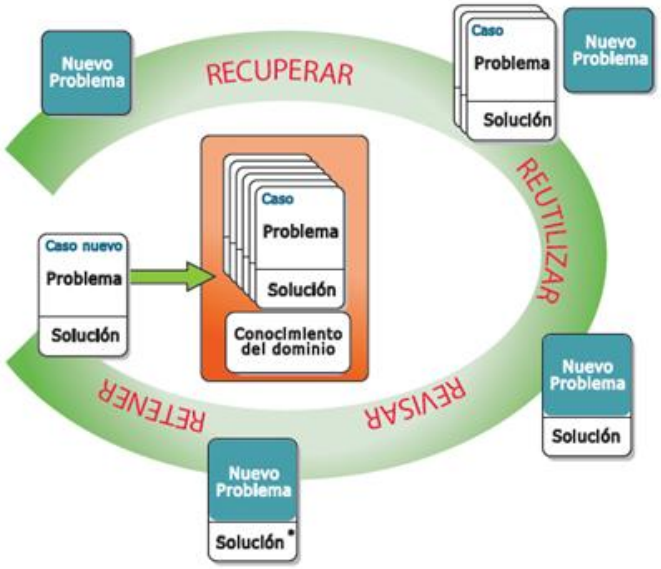


Figura 2.1: Ciclo CBR

3. **Revisar.** Revisar dicha solución y la modificamos con el fin de que se ajuste mejor al caso que queremos resolver.

4. **Recordar.** Por último, después de haber aplicado la solución con éxito, es importante recordar la solución final para añadirla, si aporta algo nuevo, como un nuevo caso a la base de casos.

Se puede ver la analogía entre los sistemas recomendadores basados en contenido y los sistemas CBR, las características de productos y preferencias de usuario que tenemos que casar se corresponden a los casos que guardan la descripción de un problema y la recomendación es por tanto, la solución de los casos. Es más, en algunos ámbitos se refieren a estos recomendadores como *recomendadores basados en casos*.

2.4. Tecnologías para el desarrollo móvil

Para el desarrollo de Madrid Live se han utilizado diversas tecnologías. Es esta sección haremos un repaso por todas ellas y señalaremos las que consideramos más destacadas.

2.4.1. Desarrollo Android

Android es una tecnología de vanguardia y altamente extendida que permite fácilmente el desarrollo de aplicaciones móviles versátiles y multiplata-

forma. Gracias a los dispositivos móviles se puede adquirir diversa información de contexto de manera transparente al usuario que se puede manejar y guardar con SQLite, el gestor de bases de datos proporcionado por Android. El *GPS*, integrado en la gran mayoría de los dispositivos móviles es un sistema de localización global el cual permite localizar lugares, al mismo dispositivo móvil y a otros dispositivos con un margen de error de 100 m. El uso de los *SDK* tan habituales en Android, da facilidades a las aplicaciones para conectarse con las redes sociales mediante las cuales se puede recopilar mucha información importante para los recomendadores de manera automática y transparente al usuario quien solo tiene que registrarse y dar así su consentimiento a la conexión con la red social.

2.4.2. Desarrollo del servidor

J2EE es una tecnología, de la capa de servicios, muy utilizada en el desarrollo de APIs que permiten la comunicación entre los clientes(p.e. aplicaciones Android) y el núcleo del servidor donde se realizan la computación de las tareas de la API. La información que manejan muchos servidores es guardada en bases de datos, como las bases de datos *MySQL* las cuales proporciona la gestión de bases de datos relacionales y multihilo y multiusuario a través de las cuales se pueden hacer consultas para recuperar información almacenada.

2.4.3. Desarrollo de sistemas de recomendación

JCOLIBRI, es un framework específicamente concebido para el diseño e implementación de sistemas CBR (razonamiento basado en casos). Al estar especialmente orientado para esta tarea ofrece muchas comodidades para diseñar el sistemas de recomendación. JCOLIBRI está desarrollado en *Java* y ofrece sus librerias y facilidades en este lenguaje que además es una de las tecnologías de programación más usada hoy en día la cual es perfecta para crear todos los componentes básicos necesarios para un sistema recomendador (realizar una recomendación, gestión de la base de datos etc.)

2.5. Conclusiones

En este capítulo se ha hecho un repaso por las técnicas utilizadas en Madrid Live. Comenzamos haciendo hincapié en la importancia actual de los sistemas de recomendación y explicamos el funcionamiento de los recomendadores tanto individuales como grupales. A continuación detallamos el funcionamiento del *trust* o valor de confianza, que es parte fundamental de nuestro modelo de recomendación grupal. Seguidamente, hemos remarcado la importancia de la información a la hora de recomendar y hemos definido el concepto de información de contexto que usamos para enriquecer y mejorar

nuestra recomendación. Por último se ha dado un repaso a las tecnologías usadas en el desarrollo de los sistemas de recomendación.

Capítulo 3

Madrid Live

Madrid Live es un sistema de recomendación de actividades de ocio para la ciudad de Madrid especialmente centrado en la recogida y uso de información social y contextual. Aunque la principal funcionalidad de Madrid Live es la recomendación de planes de ocio, también cuenta con servicios adicionales para los usuarios (por ejemplo, la creación de grupos personalizados) y otros especialmente orientados a mejorar las recomendaciones (como puede ser un servicio de geolocalización propio o la recogida de información de contexto). Para la interfaz de usuario se ha optado por la implementación de un cliente haciendo uso de la tecnología Android dados su uso y cuota de mercado actuales y las funcionalidades ofrecidas de cara al sistema (p.ej.: GPS). En la parte de *backend* Madrid Live cuenta con un servidor que se encarga de atender las peticiones del *frontend* (clientes Android).

En este capítulo ofreceremos una visión completa de las diferentes funcionalidades que ofrece Madrid Live así como de las fuentes de información empleadas a la hora de realizar recomendaciones.

3.1. Características de Madrid Live

Madrid Live es un sistema de recomendación social y contextual de actividades de ocio para la ciudad de Madrid cuya interfaz es un cliente Android (*frontend*) que realiza peticiones a un servidor web (*backend*) a través de una capa de servicios o API. Los usuarios pueden solicitar planes de ocio a través de la aplicación Android y el sistema les devolverá el que determine más apropiado en función de las restricciones impuestas. Además, Madrid Live está especialmente orientado a la recogida automática de la información que rodea al momento de la recomendación para evitar que el usuario tenga que realizar entradas manuales al sistema.

El principal objetivo del sistema es la búsqueda de combinaciones que

incluyan museos, cines, restaurantes, parques y paseos para la generación de planes de ocio, para lo que cuenta con una serie de catálogos de actividades obtenidos de varios servicios web. Además, Madrid Live trata siempre de que los planes generados encajen lo mejor posible con los gustos y preferencias de los usuarios.

Para que las recomendaciones sean lo más precisas posibles, y como cada persona es diferente, Madrid Live genera perfiles de preferencias para cada usuario que almacenan rasgos característicos de cada uno de ellos. Por ejemplo: la predilección hacia cada tipo de actividad recomendada (museos, cines, etc.), disponibilidad de desplazamiento, etc.
El principal método de actualización de estas preferencias es a través de una serie de tests que se presentan al usuario y que le permiten ayudar al sistema a conocerlo mejor. No obstante, no es el único medio para afinar las preferencias de los usuarios, ya que el sistema es capaz de analizar las acciones realizadas y determinar los gustos a partir de ellas. Por ejemplo: si a un usuario se le han recomendado museos varias veces y siempre los puntúa de forma negativa o rechaza los planes donde aparecen, el sistema es capaz de detectarlo y actualizar el perfil de preferencias para que las recomendaciones posteriores sean cada vez más precisas.

Con el objetivo de enriquecer lo máximo posible las recomendaciones, una de las principales líneas de desarrollo e investigación seguidas ha sido la recolección automática de información de contexto, es decir, todos aquellos parámetros que rodean al momento de la recomendación. Por ejemplo: el sistema detecta la fecha y hora actuales para limitar los planes al día actual o descartar actividades no disponibles en determinados días, la posición del usuario para marcarla como lugar de inicio del plan, los amigos cercanos para agregarlos automáticamente como un grupo al plan, etc.
Para todo ello se han aprovechado las funcionalidades ofrecidas por los dispositivos móviles – en concreto la tecnología Android – que, a diferencia de otros dispositivos como un ordenador personal de sobremesa, permiten recolectar gran cantidad de información de contexto. Además, Madrid Live también incluye otros módulos adicionales como un sistema de geolocalización propio basado en el GPS y las redes a las que se encuentra conectado el dispositivo que permite conocer con exactitud la posición más reciente del usuario y recuperar una lista con todos los amigos cercanos para agregarlos automáticamente al plan.

En lo que a la solicitud de recomendaciones se refiere, Madrid Live permite, por un lado, la búsqueda de planes individuales, para los que se hace uso principalmente del perfil de preferencias del usuario solicitante y de toda la información de contexto recuperada. Por otro lado el sistema también

permite la búsqueda de planes de ocio para grupos donde, a diferencia del caso individual, ha sido necesario introducir nuevos elementos para mantener niveles de satisfacción aceptables. El mayor problema al tratar con un grupo es que cada persona es diferente al resto, por lo que son necesarios mecanismos que permitan dirigir la recomendación de forma que el resultado se ajuste lo mejor posible a los gustos de todos los componentes del grupo. En concreto Madrid Live hace uso de una función de agregación propia basada en el concepto de *trust* o confianza entre los usuarios. Aquí entra en juego el factor más social del sistema y para poder generar un modelo bastante preciso de los círculos sociales de los usuarios y niveles de confianza ha sido necesario realizar la integración de Madrid Live con un servicio externo de red social (Facebook en la implementación actual) que nos permite recuperar gran cantidad de información social sobre los usuarios.

Al margen de todo lo mencionado, Madrid Live también cuenta con otras funcionalidades como un mecanismo propio de valoración de actividades que permite a los usuarios puntuar aquellos planes de ocio recomendados por el sistema y ya realizados, información que es de gran utilidad para enriquecer aún más las futuras recomendaciones para quedarse siempre con aquellas actividades que más hayan gustado a los usuarios y relegar a un segundo plano las demás.

Por último, Madrid Live incluye otras funcionalidades auxiliares como pueden ser la creación y carga de grupos de usuarios o la consulta de datos personales que contribuyen a complementar el sistema.

A continuación, a lo largo del resto de este capítulo, vamos a ir detallando todas las fuentes de información y funcionalidades que componen Madrid Live.

3.2. Fuentes de información

Cuando hablamos de sistemas de recomendación o sistemas basados en conocimiento, uno de los aspectos más importantes a tener en cuenta son las fuentes de información. Podemos modelar un sistema de recomendación muy bueno que trabaje con una serie de parámetros pero, si la información que proporcionamos al mismo es de poca calidad y variedad, las recomendaciones ofrecidas serán bastante peores de lo esperado. En general, conviene contar con catálogos que tengan una buena cantidad de elementos a recomendar y sean los más variados posibles.
En el caso de Madrid Live los catálogos empleados contienen todas las posibles actividades de ocio a recomendar, para cada una de las cuales se recopila toda la información posible tanto para ser mostrada al usuario como para

ser usada en la propia recomendación. Por ejemplo: nuestra colección de información sobre museos no sólo contiene nombres de museos sino tipos de exposiciones, direcciones, coordenadas, imágenes, etc. Toda esta información se muestra al usuario a la hora de la recomendación y le permite tener una idea mucho más clara de la actividad que se está proponiendo.

A continuación vamos a ver las diferentes fuentes de información sobre actividades de ocio empleadas en Madrid Live y su categorización en función del tipo de datos manejados.

3.2.1. Tipos de datos

Todos estos catálogos usados por Madrid Live contienen información que podemos englobar en dos bloques diferenciados:

- **Datos estáticos**: son aquellos que, una vez obtenidos, no se modifican con frecuencia. Por ejemplo: para el caso de los restaurantes tenemos un catálogo que contiene todos los disponibles en Madrid y, como la lista de restaurantes no es algo que cambie cada día, esta información no se modifica. Es decir, las recomendaciones para, por ejemplo, restaurantes, se realizan siempre partiendo de la misma base.
  Por supuesto, nada impide que cada cierto tiempo (un mes, por ejemplo) se actualice esta base de datos para incluir nuevos restaurantes.
  Entre los tipos de datos estáticos encontramos los relacionados con museos, restaurantes, parques y paseos.

- **Datos dinámicos**: en esta categoría se engloban todos aquellos datos que son susceptibles de cambiar con frecuencia. Ahora mismo solo se encuentran en esta situación los cines.
  La lista de películas y sesiones de las carteleras de cine es algo que cambia constantemente y no se puede trabajar con información antigua (ya que interferiría negativamente en las recomendaciones). Por ello, en este caso, los datos antiguos y obsoletos del catálogo de cines se eliminan y renuevan cada día para mantener siempre la información más actualizada posible.

3.2.2. Fuentes de información

Con el fin de ofrecer al usuario las mejores recomendaciones posibles, hemos tratado de seleccionar las fuentes de información más adecuadas. En este caso todos nuestros datos provienen de diferentes servicios web como Google Places o 11870.com.
Toda la información actual puede ser fácilmente ampliable para dotar a Madrid Live de una mayor variedad de elementos en los catálogos. Por ejemplo:

en vez de hacer uso solamente de los servicios actuales, podríamos incorporar otros nuevos e incrementar así la cantidad de información con la que el sistema puede trabajar.

A continuación explicaremos brevemente las fuentes de información actuales:

- **11870.com**[1]: se trata de una web donde los usuarios pueden compartir sitios de cualquier parte del mundo con opiniones, fotos y vídeos. Ofrece servicios para usuarios individuales o empresas y cuenta con una versión móvil.
  Este servicio proporciona grandes ventajas a los usuarios ya que pueden hacer búsquedas a través de la web y preparar sus planes de forma manual incluso antes de visitar una ciudad.
  La mayor utilidad para Madrid Live es que proporciona también una sencilla y cómoda API web a través de la cual se pueden realizar consultas. Por ejemplo: podemos extraer una lista de restaurantes para la ciudad de Madrid a no más de 1 km de una posición prefijada. Entre los datos devueltos encontramos información muy útil como pueden ser fotos del lugar, opiniones y valoraciones de otros usuarios o descripciones.
  De 11870.com obtenemos la información necesaria para los parques, restaurantes y paseos de Madrid. El caso de estos últimos es un poco más peculiar ya que lo que se hace es recuperar una lista de monumentos o puntos de interés de la ciudad de Madrid y generar a partir de ella paseos de no más de 2 horas uniendo algunos de los puntos de interés recuperados.

- **Google Places**[2]: es un servicio que ofrece Google a empresas y propietarios y que les permite generar fichas de sus negocios (nombre, descripción, imágenes, etc.). Una de las mayores ventajas de este servicio es que los locales inscritos en él figurarán después en Google Maps. Además, ofrece información detallada de segmentación de los usuarios que visitan sus fichas, estadísticas de visualizaciones, etc. Para los propietarios esto constituye una potente herramienta de publicidad a través de Internet, lo cual hace que Google Places sea ampliamente utilizado, principal motivo que nos ha llevado a incluir este servicio como fuente de información para la búsqueda de museos.
  Al igual que en el caso de 11870.com, incluye una API de consulta web a través de la cual podemos recuperar gran cantidad de información acerca de los tipos de exposición de cada museo, los horarios de apertura, descripciones y fotos, valoraciones de otros usuarios, etc.

[1] http://11870.com/
[2] https://support.google.com/places/?hl=es

- **guiadelocio.com**[3]: se trata de una web genérica de consulta de actividades de ocio para toda España. Aunque ofrece bastantes tipos de actividades (restaurantes, teatros, conciertos, etc.) solo la usamos para obtener las carteleras de los cines ya que no ofrece una API web.
  Para poder obtener los datos debemos realizar un análisis total de la página web (concretamente del código HTML) y recuperar de ésta la información necesaria. Almacenamos datos como la lista de cines disponibles, las sesiones y películas de cada una, actores, géneros de películas, etc.
  Como se ha mencionado antes, esta obtención de datos de guiadelocio.com se realiza una vez al día (por tratarse de información dinámica) y se limita solamente a la provincia de Madrid.

Por último mostramos en la Tabla 3.1 un resumen de las fuentes de información de empleadas en Madrid Live:

ACTIVIDAD	FUENTE DE DATOS	CATEGORÍA
Museos	Google Places	estática
Restaurantes	11870.com	estática
Parques	11870.com	estática
Paseos	11870.com	estática
Cines	guiadelocio.com	dinámica

Tabla 3.1: Fuentes de información para las actividades

3.3. El contexto y el perfil de usuario

Parte del éxito en las recomendaciones ofrecidas por cualquier sistema de recomendación procede de su capacidad de extraer toda la información útil posible acerca de los gustos o preferencias del usuario. Para ello, en muchos casos, se elaboran perfiles de preferencias o perfiles de usuario que recogen toda la información necesaria para ser usada a la hora de la recomendación. En Madrid Live también se generan estos perfiles de preferencias de usuarios, cuyo modelado se realiza a partir de toda la información recogida, que puede dividirse en tres categorías:

- **Datos estáticos**: se trata de información que cambia con poca frecuencia. Por ejemplo: los gustos generales de un usuario acerca de los tipos de exposición de un museo o su predisposición a cambiar el horario fijado para un plan.

[3] http://www.guiadelocio.com/

- **Contexto social**: gracias a la integración de una red social (Facebook en nuestro caso) es posible recuperar mucha información acerca de la red de amigos del usuario. Esto supone una gran ventaja a la hora de realizar recomendaciones en las que el usuario en cuestión forma parte de un grupo mayor.

- **Datos dinámicos**: se engloba en esta categoría toda la información de contexto que cambia con gran rapidez y que necesita ser actualizada con frecuencia. Algunos ejemplos pueden ser la posición del usuario o la hora exacta del día.

La Tabla 3.2 contiene un pequeño cuadro resumen con los datos recogidos y la categoría a la que pertenecen.

Datos estáticos	**Contexto social**	**Datos dinámicos**
Flexibilidad de horario	Amigos	Fecha
Disponibilidad para movilidad	Mensajes privados	Hora
Tipos de actividad	Fotos	Posición
Tipos de museos	Etiquetas	Amigos cercanos
Tipos de restaurantes	*follows, likes*	
Tipos de películas	etc.	

Tabla 3.2: Datos de contexto y categorías

Toda esta información se mantiene lo más actualizada posible para trabajar siempre con los datos más recientes en las recomendaciones. A continuación vamos a detallar brevemente los datos recogidos y el modelado del perfil de usuario.

3.3.1. Datos estáticos

- **Horarios**: los planes ofrecidos por Madrid Live llevan siempre asociados unos horarios para las diferentes actividades y, en general, el usuario puede seleccionar un rango horario en el que espera realizar su plan de ocio. No obstante, el sistema puede hacer cambios sobre los horarios si lo considera apropiado. Por ejemplo, si se solicita un restaurante y un parque en un rango de 16:00 a 18:00, es muy posible que Madrid Live cambie el horario y establezca un restaurante de 14:00 a 16:00 - por ser una hora más apropiada para comer - y el parque de 16:00 a 18:00.
  Aquí entra en juego un dato concreto del perfil de usuario: se trata de la disponibilidad a cambiar el horario previsto para el plan. Como cada persona reaccionará de forma diferente ante la propuesta de un plan con horario diferente al esperado, almacenamos esta información para

permitir al sistema de recomendación mayor o menor flexibilidad a la hora de alterar los horarios.
Esta información se actualiza cuando el usuario acepta o rechaza planes que están fuera del horario inicial previsto. Si los acepta, supondremos que se trata de un usuario flexible en cuanto a cambios de horario y, en caso de rechazarlos, está claro que nos encontramos ante una persona muy restrictiva respecto a los horarios.

- **Movilidad**: al igual que en el caso anterior con los horarios, ocurre algo similar con las posiciones de inicio de los planes. Los usuarios pueden elegir la zona inicial prevista donde realizar su plan pero podemos encontrarnos con personas a las que no les importará desplazarse para realizar su plan y otras que no tendrán tanta predisposición a hacerlo. De nuevo, contamos con información almacenada para conocer la predisposición del usuario a desplazarse para realizar los planes. Cuando el sistema detecte que la recomendación va orientada a un usuario o grupo al cual no le importa desplazarse, podría ampliar la zona de búsqueda de actividades para tratar de elegir alguna que podría cuadrar mucho mejor con los gustos de los usuarios aunque se encontrase algo más lejos de lo previsto.
  Como en el caso anterior, esta información del perfil de usuario se actualizará cuando el usuario acepte o rechace planes que estén lejos de la zona inicial prevista.

- **Tipos de actividad**: otro de los datos más importantes a almacenar en un sistema de recomendación de planes de ocio es la preferencia del usuario por unos tipos de actividad u otros, lo que resulta de gran importancia para poder seleccionar las mejores actividades a la hora de recomendar un plan.
  La principal aplicación de esta información es establecer un primer filtrado que se realiza antes de solicitar la recomendación. Por ejemplo: si estamos ante un usuario que tiene una predisposición nula a cambiar los horarios que espera en los planes, y que ha solicitado uno de 14.00 a 15:00 con 5 actividades incluidas - algo imposible - el sistema se verá obligado a recortar actividades para ajustarlas lo mejor posible al horario propuesto. Aquí entrarían en juego los gustos del usuario respecto a las diferentes actividades y se trataría de buscar el mejor subconjunto de las 5 solicitadas que más guste al usuario y mejor encaje en su horario.
  De forma muy similar a los casos anteriores, cuando un usuario acepta un plan en el que se incluye un tipo de actividad, se supone que esa actividad le gusta y se incrementan sus gustos por ella. En caso de rechazar el plan, se supone que esa actividad no es de su agrado y se reducen sus preferencias hacia ella.

- **Tipos de restaurantes**: aunque la carta de los restaurantes suele ser variada e incluir bastantes tipos de comida, en general, podemos catalogar los restaurantes en función de sus "especialidades" o platos más comunes. Nuestras fuentes de información (ver Sección 3.2) nos permiten obtener datos sobre el tipo de comida típico de los restaurantes que recomendamos. En concreto contamos con los tipos de comida: carne, pescado, verduras, especias-picante, exótica, guisos y pescado-marisco. Con esta información ya recogida, nos interesa, por lo tanto, conocer las preferencias de los usuarios respecto a cada tipo de comida, que se actualizan o ajustan por medio de una serie de tests como los que se detallarán en la Sección 3.3.4.

- **Tipos de museos**: de nuevo, Madrid Live almacena información sobre los diferentes tipos de exposiciones de cada museo que puede recomendar. Por ejemplo, tendremos exposiciones de pintura, arquitectura, escultura, fotografía, etc.
  Al igual que ocurría con los restaurantes, también almacenamos información acerca de las preferencias del usuario referentes a los diferentes tipos de museos, cuya actualización se lleva a cabo a través de una serie de tests que serán detallados en profundidad en la Sección 3.3.4.

- **Tipos de películas**: por último, Madrid Live también almacena información sobre los diferentes géneros de películas (acción, comedia, western, etc.).
  De nuevo, el método de actualización de estos datos es a través de un test en el que se muestran todos los tipos de películas, los cuales deben ser ordenados por el usuario de mayor a menor prioridad (se detallará en la Sección 3.3.4).

3.3.2. Contexto social

Madrid Live recupera toda la información posible y útil de la red social a la que se encuentre conectado (Facebook en nuestro caso). Entre esta información encontramos:

- **Mensajes privados**: no el contenido, sino el emisor y los destinatarios principalmente.

- **Fotos**: de nuevo, lo más interesante es recuperar las fotos subidas y su relación con otros usuarios.

- **Etiquetas o *tags*** en contenidos.

- ***Likes o follows***.

- **Relaciones de amistad**.

Con todos estos datos Madrid Live es capaz de construir el grafo social de un usuario y la red de confianza con respecto a todos sus amigos. Así se permite, por ejemplo, detectar líderes de opinión en un grupo u obtener otra información como el grado de cohesión o confianza entre los miembros de dicho grupo.

Además suponer una clara ventaja a la hora de realizar recomendaciones orientadas a grupos de usuarios, todos estos datos nos han permitido crear una función de agregación (empleada en recomendaciones grupales) que se detallará en la Sección 3.4.7.

3.3.3. Datos dinámicos

A la hora de solicitar una recomendación, el cliente Android de Madrid Live recoge todos los datos posibles acerca del contexto físico y social actual del usuario y los fija en la pantalla de recomendación. Esta funcionalidad conlleva una gran mejora en la experiencia de usuario ya que, para solicitar planes para el momento actual, no es necesario introducir ningún dato sino que Madrid Live se encargará de recoger toda la información. Por supuesto, como ya se verá en una sección posterior, el usuario puede ignorar todos estos datos prefijados y seleccionarlos libremente.

Entre la información principal de contexto tenemos:

- **Horario**: por defecto el cliente Android detecta la hora actual y fija un horario para el plan basado en ella. Además, establece los límites necesarios para garantizar la coherencia de los horarios introducidos por el usuario dado que solo se pueden solicitar planes para un solo día. Por ejemplo: si son las 14:56 el usuario no podrá elegir como hora de inicio las 10:00 de la mañana.
- **Fecha**: al igual que en el caso anterior, el cliente recoge información sobre la fecha concreta de la recomendación, lo que permite evitar realizar recomendaciones de actividades que no están disponibles ciertos días de la semana.
- **Posición**: gracias al sistema de geolocalización creado para Madrid Live, la aplicación cliente tendrá siempre la última posición conocida del usuario con la mayor precisión posible. El principal objetivo de tener la posición es poder recomendar actividades lo más cercanas posibles a la zona donde se encuentra el usuario.
- **Amigos cercanos**: de nuevo, haciendo uso del sistema de posicionamiento ya mencionado y con ayuda de la sincronización de datos con el servidor, el sistema Madrid Live es capaz de detectar qué amigos del

usuario se encuentran cerca de su posición. La principal ventaja es que, a la hora de solicitar una recomendación, no habrá que introducir manualmente los miembros del grupo sino que Madrid Live los agregará de forma automática.

3.3.4. Tests de preferencias

Uno de los principales mecanismos de Madrid Live para recopilar información acerca de los gustos del usuario son los tests de preferencias, que tienen como objetivo siempre algún tipo de actividad concreto y permiten actualizar la información de las preferencias del usuario cada vez que se realizan.

Con el fin de probar diferentes técnicas para dichos tests, se han implementado dos modelos diferenciados:

- **Algoritmo ELO**: se trata de una división del test en varias rondas en las que se presentan al usuario varios tipos de un elementos (tipos de museos en nuestro caso) y éste debe elegir los que más le gusten.

- **Listas ordenables**: en este caso se muestra al usuario una lista con todos los tipos de una determinada actividad (por ejemplo, géneros de películas) y éste debe ordenarlos de mayor a menor prioridad.

A continuación vamos a explicar el funcionamiento básico de estos tests y el sistema de valoración de actividades, información que puede ser ampliada con detalles más concretos consultando las secciones 4.1.5 y 4.1.4.

3.3.4.1. Sistema de puntuación ELO

El sistema de puntuación Elo es un método matemático, basado en cálculo estadístico usado para calcular la habilidad relativa de los jugadores de juegos como el ajedrez, el go, Dota2, etc.

Este sistema fue elaborado por el profesor de física húngaro Arpad Emrick Elo, un gran aficionado al ajedrez que desarrolló este método de cálculo de rankings para los jugadores de ajedrez considerando su nivel.
Este dato es muy importante ya que la fórmula empleada en el cálculo pondera en función de la diferencia de nivel entre los dos jugadores. Esto es, asigna menos puntuación al contrincante ganador, si es el que tiene más nivel, o por el contrario, asigna una mayor puntuación, si es el jugador de menor nivel el que gana la partida.

En 1970, la FIDE (Fédération Internationale des Échecs) acordó adoptar el sistema de puntuación Elo para clasificar a los mejores ajedrecistas de todo el mundo. Este sistema es ampliamente usado también para otros juegos pero todos de competición por parejas.
Años más tarde (Octubre de 2003) Mark Zuckerberg utilizó este sistema para realizar un sitio web llamado "Facemash" que precedió al conocido "Facebook". El propósito de esta página web era clasificar a las chicas de la universidad de Harvard según su atractivo. El programa mostraba dos imágenes de diferentes chicas y el usuario elegía cual le gustaba más. Mark Zuckerberg utilizaba el sistema de puntuación Elo para, según las respuestas de los usuarios, hacer un ranking de popularidad de las chicas.

En Madrid Live lo emplearemos para obtener un listado ordenado de los museos según las preferencias de cada usuario evitando que haya diferencias considerables entre cada tipo de museo. Es decir, utilizando y adaptando el algoritmo original, construiremos su ranking de gustos para este tipo de actividad.

Todo el cálculo matemático del sistema de puntuación Elo se basa en las fórmulas que se muestran a continuación:

$$E_A = \frac{1}{1 + 10^{(R_B - R_A)/400}} \quad E_B = \frac{1}{1 + 10^{(R_A - R_B)/400}}$$

Siendo:

- **EA:** ranking esperado para el jugador A (estimación)
- **RA:** puntuación previa (puntos acumulados hasta el momento) del jugador A.
- **EB:** ranking esperado para el jugador B (estimación)
- **RB:** puntuación previa (puntos acumulados hasta el momento) del jugador B.

Sin embargo esto solo permite obtener el valor del ranking esperado, por lo que posteriormente debe ser procesado junto con los resultados en una segunda fórmula que es continuación de la anterior.

La fórmula para generar la nueva puntuación es la siguiente:

$$Rn = Ro + C * (S - Se)$$

Dónde:

- **Rn:** es la nueva puntuación para el producto N.

- **Ro:** es la puntuación anterior (que se usó con Ra y Rb).
- **C:** constante que varía de acuerdo al tipo de cálculo.
- **S:** especifica el resultado de la elección (Tabla 3.3)

Resultado	Valor de S
Gana	1
Empata	0.5
Pierde	0

Tabla 3.3: Tabla con los distintos valores de S

- **Se:** es el ranking esperado (valor que obtuvimos en la primera fórmula).

Es importante destacar que el algoritmo explicado considera el "nivel" de cada jugador, pues, como habíamos comentado anteriormente, no se puede comparar del mismo modo una victoria o derrota (en nuestro caso) de un museo que tiene un nivel muy alto contra otro que posee una puntuación muy baja. Por ejemplo, si nos encontramos con que un usuario ha ido puntuando muy negativamente un tipo de museo que se está "enfrentando" a otro que sí ha recibido valoraciones muy positivas, la fórmula ponderaría en función de la diferencia, asignando menos puntuación al que tenía más si es que gana, o una mayor puntuación al museo peor valorado hasta el momento si resulta perdedor. Con ello se consigue que la diferencia de puntuación entre los diferentes tipos de museos esté mínimamente equilibrada.

3.3.4.2. Test de preferencias de museos: algoritmo ELO

Como ya hemos mencionado, para clasificar los diferentes tipos de museos en función de los gustos del usuario, se hace uso del algoritmo de ELO, cuyo modelo de aplicación del mismo se ha extraído del proyecto original base de Madrid Live.

Para aplicar el algoritmo, en primer lugar, hemos dividido los 12 tipos de museos en 3 grandes categorías (Tabla 3.4).

La decisión de dividir los tipos de museos en varias categorías tiene como principal objetivo poder realizar un mejor seguimiento de las opciones que se mostrarán en las 4 rondas del test final. De esta forma tratamos también de que al menos el 90 % de los tipos de exposiciones aparezcan en alguna de dichas rondas para ser puntuados.

ARTE	CIENCIA	OTROS
Pintura	Ciencias	Infantil
Arquitectura	Historia	Transporte
Escultura	Artes escritas	Deportivo
Fotografía	Curioso	Objetos

Tabla 3.4: División de tipos de exposiciones en grupos

Por último, antes de empezar con la explicación de las rondas, mencionar que todas las valoraciones de los diferentes tipos de museos están inicializadas a 0. Los valores asignados variarán entre 0 y 5, siendo esta última la puntuación más alta que podría lograr un tipo de exposición.

- ***Ronda 1*: aleatoria**
  Se muestran 6 tipos de museos aleatorios de los cuales el usuario podrá elegir hasta un máximo de 3. Una vez hecha la elección, se recalculan las nuevas puntuaciones siguiendo el algoritmo de Elo. Para ello, el enfrentamiento se produce por cada par de museos. Esto es, cada museo se enfrenta a todos los demás, considerando que un museo seleccionado por el usuario "gana" sobre uno no elegido y "empata" si ambos son o no elegidos.

- ***Ronda 2*: los 2 grupos con mayor puntuación**
  Tras haber obtenido las nuevas puntuaciones de la primera ronda entran en juego los 3 grupos (arte, ciencia y otros). Lo que trataremos de conseguir a partir de ahora, es que los museos que mostremos no sean totalmente aleatorios sino que podamos decidir qué grupos queremos mostrar, lo que permitirá hacer más fácil el descarte y encaminar en función de los gustos del usuario las siguientes rondas.
  En la ronda 2 mostraremos museos pertenecientes a los 2 grupos que más puntos obtuvieron en la ronda anterior. Es decir, se suman las puntuaciones que tienen hasta el momento todos los museos que pertenecen a cada grupo y elegimos los dos grupos con mayores puntuaciones acumuladas.

- ***Ronda 3*: más puntuado ronda 2 vs menos puntuado ronda 1**
  Esta ronda también va a depender de la anterior, ya que volveremos a sumar las puntuaciones parciales de cada grupo para elegir el mejor valorado hasta el momento por el usuario. Mostraremos tipos de museos de dicho grupo junto con tipos del grupo que no salió en la ronda

anterior, esto es, el peor valorado de la primera ronda.

- ***Ronda 4*: los favoritos**
  En esta última ronda se mostrarán al usuario los tipos de museos que más le han gustado. Es decir, los que han obtenido las puntuaciones más altas de todo el ranking establecido hasta el momento (los favoritos del usuario). Esto permitirá establecer una diferenciación aún más clara entre los tipos más votados de todo el test.

Una vez finalizadas las 4 rondas lo que tendremos es un listado ordenado de los tipos de museos según las preferencias del usuario. Es decir, habremos construido su ranking de preferencias.

3.3.4.3. Test de preferencias de restaurantes y cines

Tanto para restaurantes como para cines el test de preferencias consiste en una lista de tipos de comidas - o películas - que el usuario debe ordenar de mayor a menor preferencia. Una vez que el usuario ha ordenado todos los elementos según sus gustos, se procede a dar una valoración entre 0 y 1 a cada uno de ellos en función de la posición que ocupa en la lista ordenada y se almacenan dichas preferencias. Un tipo de comida o película valorado con 0 sería el que no llama nada la atención al usuario y el puntuado con 1 sería aquel que más agrada a éste.

En el caso de los restaurantes tenemos pocos tipos de comida, por lo que la ordenación es bastante simple. Por otro lado, contamos con gran cantidad de géneros de películas a ordenar, por lo que una de las líneas de trabajo futuro sería la sustitución del test actual de cines por otro modelo más simplificado.

3.3.5. Valoraciones de actividades

Otro de los elementos de gran utilidad para un sistema recomendador de ocio como Madrid Live son las valoraciones de los usuarios. Al margen de aquellas que puedan constar ya en las fuentes de información importadas, Madrid Live tiene un sistema de votación que permite al usuario valorar las actividades ya realizadas, lo que constituye una potente herramienta para tratar de recomendar siempre las actividades con mejores valoraciones. No nos referimos solamente al usuario que las haya votado sino a cualquier otro usuario de la aplicación.

Además, Madrid Live no sólo reconoce las valoraciones de actividades de forma individual sino también las puntuaciones de plantillas de actividades.

Por ejemplo: se ofrece a un usuario un plan de ocio que incluye una visita al Museo del Prado y un paseo por el Parque del Retiro. Cuando termina el plan, dicho usuario valora ambas actividades. Madrid Live no solo almacena esa valoración fijada para el Museo del Prado o el Parque del Retiro sino que calcula también una puntuación media para la plantilla de actividades completa, con lo que se consigue que el conjunto de estas dos actividades sea más o menos susceptible de ser recomendado a otro usuario o grupo.

La justificación está en que no todas las combinaciones de actividades son acertadas. Por ejemplo, recomendar 4 paseos consecutivos durante 8 horas sería absurdo porque el usuario se hartaría de andar. Otro ejemplo sería recomendar un restaurante para comer y acto seguido un paseo. Tal vez los usuarios, en general, prefieran un parque antes que un paseo por ser más relajado. Este mecanismo de obtención de las valoraciones de las plantillas de actividades permite al sistema adquirir conocimientos sobre estos detalles con mayor exactitud.

Por último, otra de las ventajas de este sistema es que permite a Madrid Live aprender continuamente y emplear todo ese aprendizaje en mejorar las futuras recomendaciones.

En cuanto a la forma de obtener las valoraciones, como se detallará en la Sección 4.1.4 y como se puede ver en la Figura 3.12 de la Sección 3.5, se realiza a través de una sencilla lista en la que se muestran las diferentes actividades ya realizadas y no valoradas. El usuario tiene la opción de puntuar cada una de ellas con un valor entre 0 y 5, significando 0 que no le ha gustado nada dicha actividad y 5 que le ha encantado.

3.4. Descripción funcional

Madrid Live consta de una serie de servicios que se ofrecen al usuario, algunos de los cuales son principales (como el sistema de recomendación) y otros constituyen funcionalidades más secundarias (creación y carga de grupos). Será en el módulo de recomendación donde se hará uso de toda la información de contexto y datos recogidos mencionados en las secciones anteriores.
A continuación vamos a exponer el funcionamiento de dichos servicios desde un enfoque más informativo que puede ser ampliado consultando el funcionamiento detallado en el Capítulo 4 o un modelo de casos de uso y diagramas de flujo incluido en el Apéndice C.

Más tarde, en la Sección 3.5 veremos también cómo se presentan las diferentes funcionalidades al usuario en la aplicación Android.

3.4.1. Inicio y cierre de sesión

Una vez que el usuario accede a la aplicación de Madrid Live, si no tiene una sesión activa en el sistema, se le muestra la pantalla de inicio de sesión, donde deberá introducir sus credenciales de acceso. Dado que el sistema está integrado con la red social Facebook, el inicio de sesión se lleva a cabo a través de sus servicios.

Cuando las credenciales de acceso han sido verificadas correctamente, se almacenan junto con los datos de sesión en la aplicación Android y se crea una nueva sesión para el usuario en el servidor. Acto seguido se permite el acceso del usuario al resto de funcionalidades de Madrid Live que no pueden ser usadas sin una sesión activa (recomendación, geolocalización, etc.).

En principio, aunque las sesiones son permanentes sin importar que se esté haciendo uso de la aplicación o no, el sistema comprueba siempre la validez de las sesiones activas y las hace caducar si no se ha detectado actividad en mucho tiempo o las renueva cuando sea necesario. En el caso de la caducidad, se anularía la sesión en curso y se obligaría al usuario a realizar un *login* de nuevo.

Por otro lado, cuando un usuario ya tiene una sesión activa, siempre cuenta con la opción de realizar un cierre de sesión o *logout* manual.

3.4.2. Recuperación de datos de la red social

Con el fin de mejorar las recomendaciones realizadas, sobre todo a grupos de usuarios, se ha integrado Madrid Live con una red social (Facebook). Gracias a ello contamos siempre con gran cantidad de información adicional acerca de los usuarios y sus círculos sociales, lo que nos permite establecer niveles de confianza entre usuarios y detectar líderes de opinión en los grupos.

Tras un inicio de sesión, o si han pasado más de 24 horas desde la última sincronización, se recogen de nuevo los datos del usuario de la red social y se sincronizan con el servidor para trabajar siempre con la información más reciente posible.

Toda esta carga de datos se lleva a cabo siempre de forma completamente transparente al usuario para que no afecte a la experiencia de uso de la aplicación. Además, solamente se recuperan aquellos datos que realmente son necesarios o pueden aportar beneficios al proceso de recomendación, nunca información privada y personal de los usuarios. Por ejemplo: se sincronizan los mensajes privados del usuario, pero de ellos solamente tomamos el emisor, el receptor y la fecha, nunca el contenido, ya que no aportaría mucho al

sistema de recomendación y se trata de información sensible. Por otro lado, conocer las interacciones de un usuario con sus amigos sí es algo importante para establecer niveles de confianza.

Podemos encontrar información más precisa acerca de la recogida de datos de la red social en las secciones 4.1.2 y 4.3.4 donde se explica en detalle el procedimiento de sincronización de información de la red social con el servidor y se muestran los datos concretos extraídos.

3.4.3. La geolocalización

Otra de las funcionalidades incorporadas al proyecto base de Madrid Live ha sido el sistema de geolocalización, que permite obtener información muy precisa y actualizada de las útlimas posiciones conocidas de los usuarios. Además, gracias a este módulo también obtenemos una ventaja adicional que es la de conocer qué amigos se encuentran cerca a la hora de la recomendación para poder agregarlos automáticamente al plan.

Este sistema de geolocalización, que requiere del uso del GPS o que el usuario esté conectado a alguna red, se arranca tras un inicio de sesión, monitoriza a intervalos aceptables la posición del usuario y se encarga de sincronizar esta información con el servidor.

Para mantener la privacidad, siempre se ofrece al usuario la posibilidad de detener esta recogida de datos y sincronización con el servidor, lo que tiene como efecto lateral que no se podrán detectar automáticamente los amigos cercanos ni la posición inicial del plan.

En caso de ser necesaria más información acerca del funcionamiento del módulo de geolocalización, puede consultarse la Sección 4.3.5, en la que se explica en detalle la recogida de datos mediante GPS y su sincronización con servidor.

3.4.4. Los tests de preferencias

Unos de los mecanismos para recopilar información acerca de las preferencias de los usuarios que ya se han mencionado en la Sección 3.3.4 son una serie de tests acerca de diferentes tipos de museos, restaurantes o películas que, una vez realizados, permiten actualizar el perfil de preferencias de los usuarios. Estos tests son completamente opcionales y pueden ser repetidos continuamente.

Cada vez que el usuario los rellena, se actualizan sus gustos en función de las respuestas ofrecidas y, aunque esto constituye una potente fuente de

información, Madrid Live cuenta con otros mecanismos automáticos para deducir los gustos de los usuarios. Por ejemplo: si en un principio el sistema ha deducido que a un determinado usuario le gustan las películas de acción mucho más que cualquier otro tipo, pero después este usuario rellena el test de preferencias de películas y en él pone al mismo nivel las películas de acción que las de terror, entonces el sistema reajustará el perfil de preferencias para reducir la importancia del género cinematográfico de acción para nivelarlo correctamente.

En resumen, los tests de preferencias son una herramienta ofrecida al usuario que, aunque no es imprescindible para un correcto funcionamiento del sistema, sí es de gran ayuda para afinar al máximo los perfiles de preferencias.

Para ofrecer más variedad, cada uno de los tests sigue un modelo diferente:

- **Museos**. El test consta de 4 rondas, en cada una de las cuales se muestran al usuario 6 tipos de exposiciones (pintura, escultura, ciencia, etc.) de las cuales el usuario debe elegir entre 1 y 3 de las que más le atraigan.
  Este test está basado en el algoritmo ELO (ver Sección 3.3.4) para hacer más amigable su realización y, a la vez, equilibrar mejor los pesos dados a cada tipo de exposición.
  Los elementos mostrados en las rondas no son completamente aleatorios sino que siguen un cierto patrón. En la primera ronda, por ejemplo, sí se muestran 6 tipos de museos completamente aleatorios (dado que no hay información inicial), mientras que en las rondas 2 y 3 se alternan elementos de los mejor valorados con los peor valorados. Por último, en la ronda 4 se muestran las 6 opciones más elegidas por el usuario para establecer una diferenciación más clara entre ellas.

- **Restaurantes y cines**. En ambos casos el test tiene un funcionamiento muy similar: se presenta al usuario una lista de géneros cinematográficos / tipos de comidas que deben ser ordenados de mayor a menor orden de preferencia.
  En función de la lista resultante, se asignarán mayores o menores pesos dentro del perfil de preferencias del usuario.

Podemos encontrar nueva información acerca del funcionamiento detallado de estos tests en la Sección 4.1.5, donde se abordan el diseño e implementación concretos de los mismos.

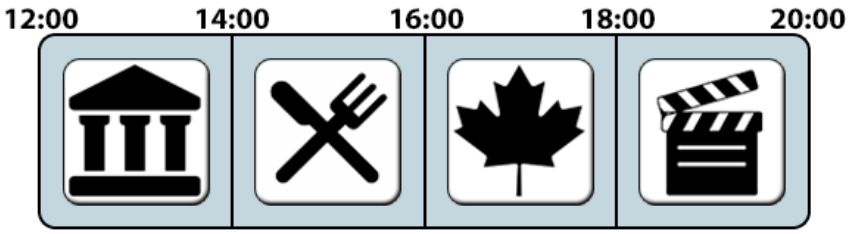


Figura 3.1: Ejemplo de plantilla abstracta

3.4.5. Gestión de grupos

Con el fin de agilizar el proceso de solicitud de un plan de ocio y mejorar así la experiencia de usuario se ha introducido la funcionalidad de creación de grupos personalizados de usuarios, que no son más que una lista de amigos bajo un determinado nombre. Por ejemplo: si uso Madrid Live con frecuencia para buscar un plan para mi familia, puedo crear un grupo denominado "Familia" que incluya a todos los miembros con los que suelo realizar actividades.

El usuario podrá, por tanto, crear grupos según considere oportuno, algo que deberá hacer a través de la pantalla de recomendación de la aplicación Android (ver Sección 3.5.10). En ella, una vez que se han agregado los usuarios deseados al plan, se da la opción de "guardar grupo" que solicitará el nombre del nuevo grupo y, tras comprobar su validez, lo almacenará en el sistema para futuros usos.

Además, como se verá en la Sección 3.5.10, la lista con los grupos creados hasta el momento por el usuario puede ser consultada en todo momento a través de la sección de grupos de la aplicación cliente, que mostrará tanto el nombre del grupo como los usuarios que lo componen. A través de esta misma pantalla se da también la opción de eliminar grupos previamente creados así como la opción de refrescar la lista (sincronizarla con el servidor) en caso de que la información visualizada no sea la correcta.

3.4.6. Proceso de recomendación basado en plantillas

Las generación de planes de ocio y recomendaciones de actividades son las funcionalidades principales de Madrid Live y por ello requieren una atención especial. En los módulos de recomendación se reúne y procesa toda la información recopilada por el sistema (contexto, perfil de usuario, fuentes de información, etc.) para tratar de ofrecer siempre el mejor plan de ocio posible.

En Madrid Live hemos elegido un modelo de generación de planes de

Figura 3.2: Ejemplo de plantilla concreta

ocio basado en plantillas de actividades. Tenemos, por un lado, las plantillas abstractas (ver Figura 3.1) que no son más que un conjunto de actividades no definidas que conforman un plan de ocio. Se podría ver como un molde genérico de actividades. Por ejemplo: de 16:00 a 20:00 una visita a un parque (sin definir cual) seguida de una exposición en algún museo.
Por otro lado nos encontramos las plantillas concretas que, como se ve en la Figura 3.2, son un conjunto de actividades definidas que conforman un plan de ocio (generadas a partir de una plantilla abstracta). Por ejemplo: de 16:00 a 20:00 una visita al Parque del Retiro seguida de una exposición en el Museo Reina Sofía.

El usuario puede solicitar un plan de ocio a través de la pantalla de recomendación de la aplicación Android (ver Sección 3.5.5) en la que se le presentan todas las opciones disponibles para establecer una configuración deseada para el plan de ocio resultante.

En primer lugar podrá seleccionar si la solicitud se realiza para una persona o para un grupo. Por defecto Madrid Live detecta todos los amigos cercanos al solicitante y los agrega al plan actual, pero el usuario puede configurar libremente las personas que compondrán el grupo. También existe la posibilidad de crear o cargar grupos personalizados como se ha visto en el apartado anterior.
En general, salvo que el grupo de personas difiera bastante de lo esperado, Madrid Live ofrece funcionalidades para que se pueda configurar de forma completamente automática o en el menor número de pasos posibles.

A continuación se da la opción al usuario de elegir tanto la hora como el lugar donde se espera realizar el plan de ocio. De nuevo, por defecto el sistema recoge la información de contexto y fija como posición de inicio la actual del usuario (aquí entra el sistema de geolocalización), como hora inicial la actual y deja un margen de 5 horas para la realización del plan. Estas opciones pueden ser modificadas a través de la aplicación Android antes de

solicitar el plan, con las únicas restricciones de que el horario debe cuadrar con el día actual y la posición no debería alejarse demasiado de Madrid.

Por último, el usuario debe elegir qué actividades de las 5 posibles (cines, parques, paseos, museos y restaurantes) le gustaría que apareciesen en el plan final.

Una vez recogida toda esta información, y tras haber solicitado el plan el usuario, se envía al servidor para elaborar una configuración del plan con las posibles restricciones.
El primer paso realizado tras la recepción de dicha información consiste en un filtrado muy simple sobre las actividades esperadas para el plan en función de las duraciones esperadas para las mismas. Por ejemplo: si se ha dado un margen de 2 horas y se han solicitado los 5 tipos de actividades, habrá que recortar algunas para que el plan encaje lo mejor posible.
Aquí entra en juego el perfil de preferencias del usuario, ya que permite tomar decisiones como: alargar o mantener el horario del plan, mover o mantener la posición de inicio, elegir qué actividades entrarán en el plan tras el recorte y cuáles no, etc. Si la recomendación fuese para un grupo entero, se considerarían las medias de los respectivos perfiles de preferencias y no solo las del usuario solicitante.

Con todas las restricciones ya analizadas y bien definidas, se procede a buscar un plan de ocio que encaje con las mismas dentro de los ya realizados por otros usuarios (plantillas concretas ya existentes), lo que resultará mucho más ventajoso que crear uno nuevo dado que ya tendremos información adicional más fiable (por ejemplo, las valoraciones de estos usuarios). Aquí entraría en juego el denominado "recomendador de plantillas concretas" que se encargaría de buscar el mejor plan de ocio disponible entre todos los ya realizados como se muestra en la Figura 3.3.

Si no se encontrase un plan que encajase suficientemente bien con las preferencias del usuario/grupo y las restricciones impuestas, entonces se pasaría a solicitar uno completamente nuevo (generación de una plantilla abstracta) ya que sería la única forma de conseguir un plan mejor. De esta funcionalidad se encargaría el módulo que hemos denominado "recomendador de plantillas abstractas", que haría uso de una base de datos con todas las posibles configuraciones de planes de ocio y de los recomendadores individuales de cada tipo de actividad (cine, museo, etc.) para generar la plantilla concreta que se ofrecerá como recomendación (ver Figura 3.4).

En todos los recomendadores, cuando estamos ante una recomendación grupal, se buscan actividades o planes para los diferentes miembros del gru-

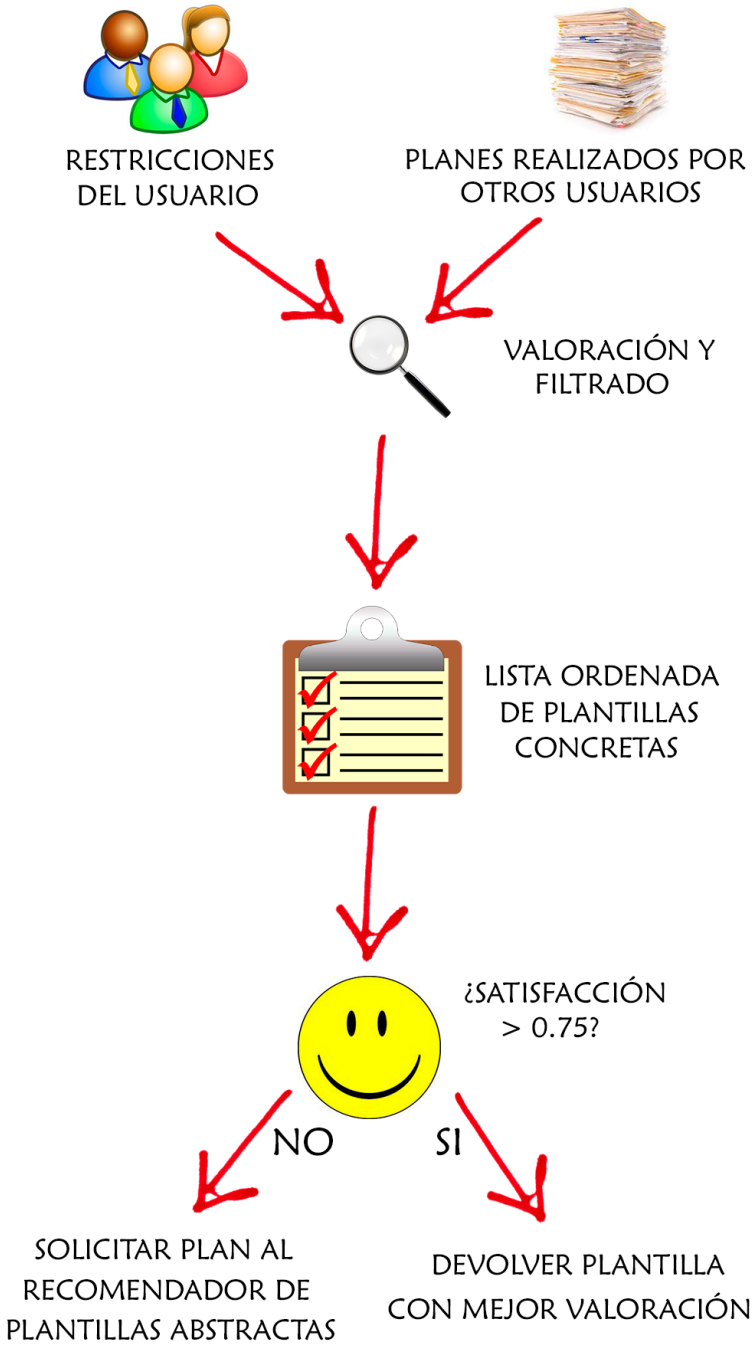


Figura 3.3: Esquema de recomendación de plantillas concretas

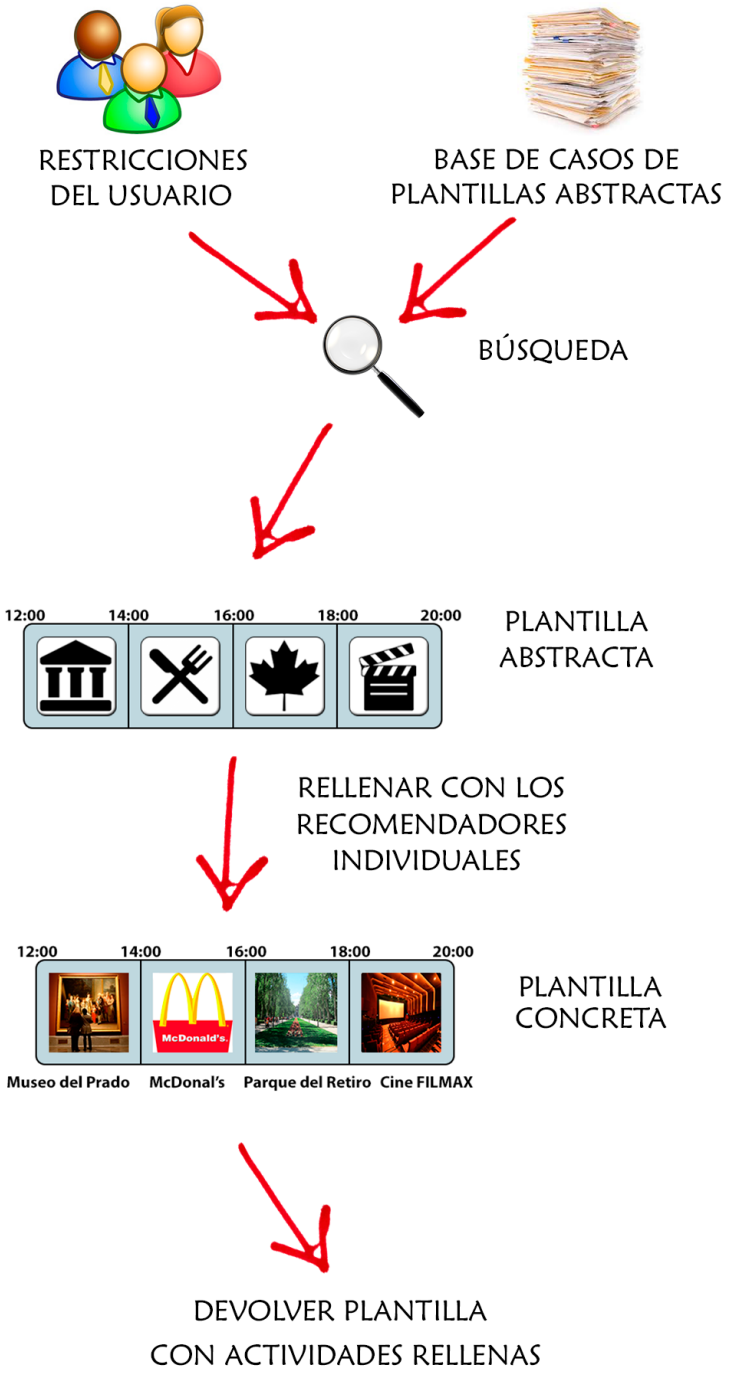


Figura 3.4: Esquema de recomendación de plantillas abstractas

po y se filtran para obtener un único resultado haciendo uso de funciones de agregación. En el caso de Madrid Live, la integración de Facebook nos ha permitido elaborar una función de agregación propia basada en las relaciones de confianza de los usuarios cuya implementación concreta puede ser consultada en la Sección 3.4.7.

Tanto si se ha encontrado un buen plan entre los ya realizados por otros usuarios, como si se ha tenido que crear uno nuevo de cero, se devuelve al usuario solicitante para ser mostrado en la aplicación Android (ver Sección 3.5.6). Cada tipo de actividad muestra toda la información posible y disponible acerca de la misma. Por ejemplo: un restaurante podrá tener un enlace a la web del mismo, fotos, opiniones de otros usuarios, etc. mientras que en un paseo lo más relevante serían los puntos de interés por los que pasará el usuario.

Cuando se ha terminado de revisar el plan, el usuario tiene la opción de aceptarlo o rechazarlo. En el último caso, se le enviaría de nuevo a la pantalla de recomendación para solicitar uno nuevo. Por otro lado, si lo acepta, se agrega el plan al historial del usuario o grupo y se considera que lo va a llevar a cabo.

Aunque ésta es la principal vía para solicitar recomendaciones, Madrid Live también ofrece algunas actividades genéricas que el usuario puede consultar en cualquier momento en la pantalla principal del cliente Android (ver Sección 3.5.2). En concreto, se muestran los 5 lugares más visitados por otros usuarios y el mejor valorado de todos, de forma que el usuario pueda tener de un vistazo alguna idea rápida de actividad a realizar.

Además, tras haber aceptado y realizado algún plan, el usuario tiene la opción de valorar cada una de las actividades del mismo, lo que permite al sistema mejorar continuamente el conocimiento o información acerca de las actividades de ocio a recomendar. Estas valoraciones de usuarios son, junto con la información de contexto y los perfiles de preferencias, los datos más importantes para ofrecer las mejores recomendaciones posibles no solo al usuario que ha valorado las actividades sino a cualquiera de Madrid Live. Para aprovechar al máximo este mecanismo de puntuación, también se detectan las respuestas del usuario para actualizar su perfil de preferencias en función de las mismas. Por ejemplo: si un usuario valora siempre negativamente los paseos, esta información se guardará en su perfil de preferencias para evitar recomendárselos siempre que sea posible.

Todo el algoritmo de recomendación comentado queda recogido de forma mucho más esquemática y clara tanto en la Figura 3.5 como en los puntos

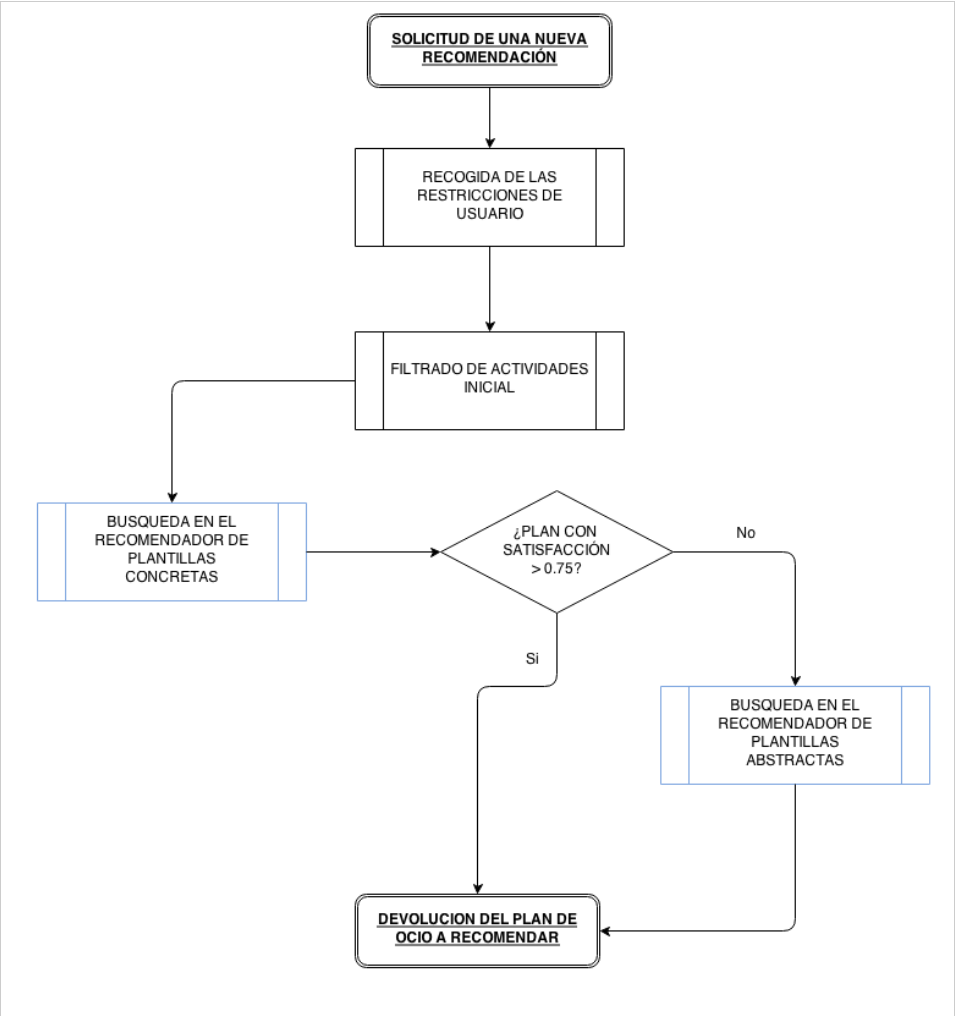


Figura 3.5: Esquema del algoritmo de recomendación completo

que detallamos a continuación:

- ***Recogida de restricciones.***
  En primer lugar, a través de la capa de servicios se recogen las restricciones impuestas por el usuario. Por ejemplo, el horario esperado para el plan, la posición inicial o la lista de actividades que le gustaría que apareciesen en él.

- ***Filtrado inicial.***
  Se realiza a continuación un filtrado sobre el número y tipo de actividades que se incluirán en el plan resultante. Se combinan para ello la predisposición del usuario a realizar planes fuera del horario señalado y sus preferencias por cada tipo de actividad. Por ejemplo: si un usuario da un margen de 2 horas para un plan, marca los 5 tipos de actividad para ser realizados en ese intervalo y, además, a dicho usuario no le gusta que se le recomienden planes fuera del horario previsto; entonces el sistema recortará las 5 actividades solicitadas y dejará solo una de ellas - la preferida del usuario -.
  Por otro lado, si otro usuario marca un intervalo de 2 horas para su plan con 5 actividades pero a éste no le importa que el plan propuesto se salga del horario, Madrid Live no recortaría ninguna actividad y alargaría el plazo del plan previsto.

- ***Recomendador de plantillas concretas.***
  Acto seguido se procede a buscar una plantilla que concuerde con las restricciones dentro de las ya realizadas por otros usuarios. Para ello se hace uso del recomendador de plantillas concretas mencionado anteriormente.
  Se lleva a cabo esta primera acción ya que conviene buscar primero opciones entre los planes de ocio ya realizados porque siempre tendrán más información que cualquier plan generado de cero (por ejemplo, tendremos valoraciones de usuarios para estas plantillas).
  Si el plan solicitado fuese para un grupo, se aplicaría la función de agregación basada en el *trust* que veremos en la Sección 3.4.7.
  Si no se encontrase ningún plan concreto apropiado, se pasaría a solicitar uno al recomendador de plantillas abstractas.

- ***Recomendador de plantillas abstractas.***
  A este punto solamente se llegaría si no se ha encontrado un plan de ocio realizado por otros usuarios que encaje con la nueva solicitud.
  Aquí se generaría una nueva plantillas de actividades abstractas haciendo uso del recomendador de plantillas abstractas y se asignaría un horario para el plan completo.
  A continuación se haría uso de los recomendadores de actividades individuales para rellenar cada uno de los huecos de la plantilla abstracta

generada anteriormente.
Si el plan solicitado fuese para un grupo, se aplicaría la función de agregación basada en el *trust*˙
Tras haber rellenado cada hueco de actividad con la correspondiente concreta, almacenaría el plan completo para ser devuelto.

- ***Devolución de la recomendación.***
  Tanto si se ha hecho uso del recomendador de plantillas concretas como si se ha tenido que solicitar un nuevo plan al recomendador de plantillas abstractas, se recoge el plan de ocio resultante y se devuelve al usuario, de nuevo, a través de la capa de servicios (API).

3.4.7. Recomendación grupal social

En el caso de las recomendaciones grupales cobra especial importancia la recuperación de información de los círculos sociales de los usuarios proporcionada por la red social con la que se integra Madrid Live. A continuación veremos cómo manejamos toda esta información para establecer niveles de *trust* o confianza entre los usuarios y lo aplicamos a las funciones de agregación clásicas para mejorar las recomendaciones a grupos.

3.4.7.1. Cálculo del valor de *trust*

Para establecer niveles de confianza entre los usuarios de Madrid Live se hace uso de un parámetro que hemos denominado *trust* o confianza. Entre cada dos usuarios (A y B) dados se calcula un valor de *trust* que varía entre 0.1 y 1.0, significando 0.1 que la relación de confianza entre A y B apenas existe y 1.0 que se trata de una relación muy sólida.

Es importante tener en cuenta que, como se comprobará a continuación, el valor del trust no es simétrico. Es decir: la confianza entre A y B no tiene por qué ser la misma que entre B y A. Dependerá, en general, de las interacciones que se hagan entre A y B y viceversa. Lo que sí es de esperar es que los valores de confianza no difieran demasiado.

Este valor de *trust* será usado para ponderar las recomendaciones cuando se trate de grupos de usuarios. Se establecerán métodos de selección guiados por este valor de *trust* que ayuden a recomendar las actividades más apropiadas para el grupo en cuestión.

El cálculo del *trust* se hace partiendo de un modelo propuesto en otro proyecto (Lara Quijano Sánchez, 2010) y el valor final vendrá determinado por 4 parámetros adicionales denominados: *intimacy*, *intensity*, *duration* y *reciprocal services*.

Para poder calcular dichos elementos es necesario obtener previamente una serie de datos de algún servicio externo, que en nuestro caso se trata de la red social a la que se conecta Madrid Live (Facebook). Dicha información se mantendrá siempre lo más actualizada posible.

Entre los datos necesarios tenemos:

- Mensajes privados enviados.
- Fotos subidas a la red social y etiquetas en las mismas (*tags*).
- Mensajes escritos en el muro del usuario, páginas o eventos.
- Mensajes de estados actualizados.
- Comentarios en los diferentes elementos (fotos, *posts*, etc.).
- *Likes* o *follows* del usuario a los diferentes elementos (fotos, *posts*, comentarios, etc.).

A continuación se explica cómo se obtiene cada uno de los 4 factores que intervienen en el cálculo del *trust* para dos usuarios dados u y v:

Intimacy

- **A1**) Calcular el total de fotos de u en los últimos D días (la unión de las fotos en las que está etiquetado u y las subidas por u).
- **A2**) Calcular el número de fotos en las que aparecen etiquetados u y v en los últimos D días.
- **A3**) Calcular el número de fotos subidas por u en los últimos D días en las que aparece etiquetado v pero no u.
- **A4**) Calcular un parámetro denominado "per_tags" que será: $(A2 + A3)/A1$.
- **A5**) El valor de *intimacy* vendrá determinado por la Tabla 3.5.

Intensity

- **B1**) Contar cuántas veces ha existido una interacción diaria (ID) entre u y v de media en los últimos D días (actualmente 3 meses). Para que exista una interacción diaria, u ha tenido que enviar un mensaje a v, publicar algo en su muro, comentar el estado de v, etc.

Per_ tags	***Intimacy***
>0.75	1.0
>0.5	0.7
>0.25	0.5
>0.1	0.3
<0.1	0.1

Tabla 3.5: Correspondencia entre *intimacy* y "per_tags"

Interacciones diarias (media)	***Intensity***
1 interacción diaria	1.0
>= 36 IDs totales	0.7
>= 12 IDs totales	0.5
>= 3 IDs totales	0.3
<3 IDs	0.1
Ninguna ID	0.0

Tabla 3.6: Correspondencia entre *intensity* e interacciones diarias

- **B2**) El valor de *intensity* vendrá determinado por la Tabla 3.6.

Duration

- **C1**) Calcular el número de amigos comunes entre u y v.
- **C2**) El valor de *duration* vendrá determinado por la Tabla 3.7.

Amigos comunes	***Duration***
>25	1.0
>15	0.7
>10	0.5
>5	0.3
<5	0.1

Tabla 3.7: Correspondencia entre *duration* y el número de amigos comunes

Reciprocal services

- **D1**) Calcular el número de *likes* de u.
- **D2**) Calcular el número de *likes* comunes entre u y v.

- **D3**) Calcular el valor intermedio denominado "per_likes" como: $D1/D2$.
- **D4**) El valor de *reciprocal services* vendrá determinado por la Tabla 3.8.

Per_likes	***Reciprocal services***
>0.75	1.0
>0.5	0.7
>0.25	0.5
>0.1	0.3
<0.1	0.1

Tabla 3.8: Correspondencia entre *reciprocal services* y "per_likes"

Cálculo del valor de trust

Una vez calculados todos estos parámetros, el valor de *trust* entre dos usuarios u y v vendrá determinado por la fórmula:

$Trust(u,v) = 0{,}361 \times intimacy(u,v) + 0{,}239 \times reciprocal\_services(u,v) + 0{,}219 \times intensity(u,v) + 0{,}181 \times duration(u,v)$

Las constantes (pesos) que acompañan a cada uno de los factores del *trust* responden a la importancia dada para cada uno de ellos en la determinación del grado de confianza de una persona con otra. La configuración actual ha dado buenos resultados en las pruebas pero, por supuesto, es susceptible de ser modificada con libertad.

3.4.7.2. Función de agregación: *Trust Weighted Mean*

Cuando estamos frente a recomendaciones para grupos de usuarios, el procedimiento a seguir es solicitar recomendaciones individuales para cada uno de los miembros y después seleccionar la más apropiada para el grupo de entre todos los resultados.

Para elegir la mejor opción, se hace uso de funciones de agregación que se encargan de ponderar todos los elementos recomendados para poder ordenarlos en función de cómo resulten de apropiados para todo el grupo.

Algunas de las funciones de agregación clásicas son:

- ***Media:*** cada vez que se agrega una actividad al conjunto de actividades recomendadas se va calculando la media de "satisfacción". Es decir,

se suma la "satisfacción" de cada usuario a cada actividad y se divide por el número total de actividades. Una vez que se han añadido todas las actividades se devuelve la que tenga mayor media.
Con esta función se consigue que todos los usuarios queden satisfechos de la manera más igualada posible.

- ***Mínima miseria:*** en este caso se pretente localizar la actividad que menos disguste al grupo en general. Para ello, cada vez que se añade una actividad al conjunto, se guarda el grado de "satisfacción" del usuario que menor satisfacción tenga para esa actividad. Cuando ya se han agregado todas las actividades, se toma aquella con el máximo de los grados de satisfacción almacenados y se devuelve como solución.
El objetivo de esta función de agregación es que ningún usuario quede totalmente descontento con el plan.

- ***Máxima:*** esta función de agregación guarda el grado de satisfacción del usuario que tenga la "máxima satisfacción" por esa actividad. Una vez que se han añadido todas las actividades se devuelve aquella que tenga el mayor grado de satisfacción guardado.
En este caso lo que se consigue es que un usuario en concreto esté lo más satisfecho posible sin tener en cuenta el resto, lo cual podría ser útil si, por ejemplo, este usuario es el más influyente del grupo.

En Madrid Live hemos creado una nueva función de agregación (que hemos denominado *Trust Weighted Mean*) partiendo de la media clásica e incorporando el valor de *trust* visto en la Sección 3.4.7.1. Hemos elegido la agregación media clásica como punto de partida por considerar que se trata de la más equilibrada y justa de todas. Como se ha mencionado antes, ésta consigue que la satisfacción media del grupo sea la mejor posible.

Para comprender la incorporación del *trust* a las agregaciones de recomendaciones grupales pasaremos primero por detallar un poco la función de agregación media.

Partimos de un grupo de usuarios U con miembros $u_1...u_N$. También de una serie de recomendaciones individuales $(r_1...r_M, M \leq N)$ hechas para estos usuarios. Cada recomendación tendrá una valoración (valor de similitud) $v_1...v_M$.
En los casos en los que una misma actividad r_i se haya recomendado para varios usuarios $(u_i...u_j)$ su valor de similitud será:

$$v_i = \frac{\sum val(r_i, u_x)}{|u_i...u_j|}$$

Donde $val(r_i, u_x)$ es el valor de la valoración de la actividad r_i para el usuario u_x. Es decir: la valoración final de esa actividad concreta será la me-

dia de todas las obtenidas para dicha actividad para los diferentes usuarios.

Una vez estén las M actividades valoradas siguiendo este método, se tomará como recomendación final aquella que más valoración media haya tenido.

De forma general: dado un grupo G y una actividad A, la valoración grupal para dicha actividad (*grec*) vendrá dada a partir de las valoraciones individuales (*srec*) siguiendo la fórmula:

$$grec(G, A) = \frac{1}{|G|} \sum_{u \in G} srec(u, A)$$

En nuestro caso partimos de esta última fórmula de la agregación media clásica para incluir el valor del *trust*.
Para ello, es importante tener en cuenta que los valores de confianza (*trust*) generados se encuentran siempre entre 0.1 y 1.0, lo cual nos permitirá usarlos a modo de peso o ponderación para las diferentes recomendaciones.

Antes de continuar, vamos a definir el *trust* entrante de un usuario dentro de un grupo (*itrust*) como el valor de confianza medio que tienen todos los usuarios del grupo en él. Es decir, dado un grupo de más de un usuario G y un usuario u perteneciente a G ($u \in G$), calcularemos este *trust* entrante como:

$$itrust(G, u) = \frac{1}{|G|-1} \sum_{v \neq u \wedge v \in G} trust(u, v)$$

Donde trust(u,v) representa el valor de confianza del usuario v hacia el usuario u (ver cálculo en la Sección 3.4.7.1).

Teniendo esto en cuenta, y recordando que el valor de itrust(G,U) siempre estará entre 0.1 y 1.0, lo aplicamos a la agregación media de la siguiente forma:

$$grec(G, A) = \frac{1}{|G|} \sum_{u \in G} itrust(G, u) \times srec(u, A)$$

Es decir, hacemos que el valor del *trust* entrante dentro del grupo sirva como peso para la recomendación de ese usuario.

Con esto lo que conseguimos es que las personas con más confianza por parte del grupo – líderes de opinión[4] – tengan más peso a la hora de la elección final de la actividad. Aquí es donde realmente cobra importancia

[4]http://es.wikipedia.org/wiki/L%C3%ADder_de_opini%C3%B3n

el parámetro de *trust* o confianza, ya que estos líderes de opinión tienen un impacto mucho mayor en el grupo a la hora del plan de ocio que se debe recomendar.
Además, en general, también logramos que los componentes más unidos del grupo salgan lo más satisfechos posible - por tener más en cuenta sus valoraciones - frente a personas que se hayan agregado al grupo de forma puntual. Es decir: favorecemos al núcleo del grupo.

3.5. Descripción de la interfaz

Tras haber visto de forma básica todas las funcionalidades ofrecidas por el sistema estamos en disposición de detallar la presentación de cada una de ellas en la aplicación cliente Android.

3.5.1. Inicio de sesión

Madrid Live está integrado con Facebook, lo que significa que no será necesario registrarse en la aplicación sino que bastará con que el usuario haga uso de su cuenta de Facebook habitual.

Por lo general, si el usuario ya tiene la aplicación de Facebook instalada en su dispositivo con una sesión abierta, Madrid Live hará uso de la misma. Por lo tanto, en estos casos, el login debería ser automático (sin introducir datos) tras pulsar el botón (Figura 3.6).

Como ya se ha comentado, la integración con una red social (Facebook en este caso) nos permite recopilar gran cantidad de información acerca del usuario y su círculo social para mejorar todo lo posible las recomendaciones ofrecidas.

3.5.2. Pantalla principal

Tras iniciar sesión en el sistema se le presentará al usuario la pantalla principal (Figura 3.7), que contendrá una serie de recomendaciones básicas basadas en las actividades realizadas por otros usuarios. En concreto, se mostrarán las cinco actividades más realizadas por los usuarios y la mejor valorada de todas.
Estas recomendaciones iniciales tienen el propósito de mostrar al usuario algún "plan" rápido a realizar sin tener que avanzar más en la aplicación. En el caso de estas sugerencias iniciales no se hace uso de ninguna información del perfil del usuario concreto sino que se trata de recomendaciones genéricas. Además, esta pantalla ofrecerá al usuario un botón a modo de acceso directo a la sección de recomendación.

Figura 3.6: Madrid Live: inicio de sesión

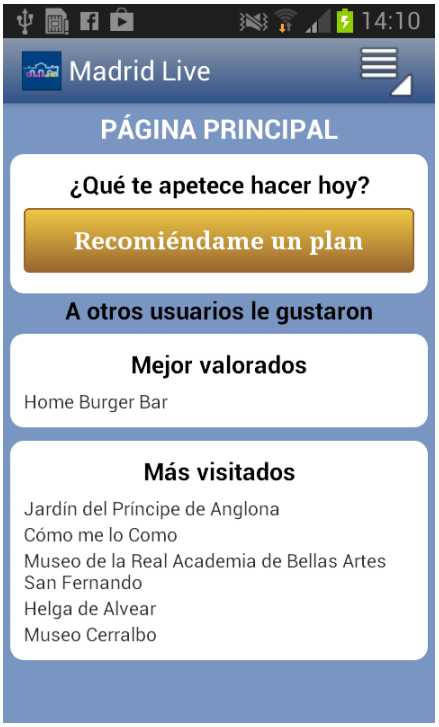


Figura 3.7: Madrid Live: pantalla principal

3.5.3. Menú principal

La aplicación Android de Madrid Live dispone de un menú principal (Figura 3.8) para acceder a las diferentes pantallas o secciones que puede ser desplegado haciendo uso del icono superior derecho de la pantalla (ver Figura 3.8).

Entre las opciones que podemos encontrar en dicho menú tenemos:

1. **Principal**: da acceso a la pantalla principal del sistema.

2. **Recomendador**: muestra la pantalla o sección de recomendación donde el usuario puede configurar el plan de ocio a solicitar.

3. **Valorar**: mediante esta opción el usuario accede a la pantalla de valoración de actividades ya realizadas.

4. **Mejorar preferencias**: permite acceder a los diferentes tests de preferencias, que pueden ser realizados tantas veces como el usuario desee y permiten crear un perfil más refinado de los gustos del usuario.

5. **Mis datos**: sencilla sección que muestra una serie de datos personales básicos del usuario con la sesión activa.

6. **Mis grupos**: muestra la pantalla donde se visualizan los grupos creados por el usuario.

7. **Iniciar/Detener geolocalización**: permite activar/desactivar la geolocalización – y sincronización con servidor –.

8. **Cerrar sesión**: cierra la sesión activa y elimina las credenciales del sistema.

3.5.4. Sistema de geolocalización

Madrid Live cuenta con un sistema de geolocalización propio que hace uso del GPS y las redes del dispositivo. Su principal uso es determinar la posición exacta del usuario y sus amigos y agregarla como información de contexto para las recomendaciones.

Este sistema de posicionamiento puede activarse o desactivarse a través del menú principal (Figura 3.9). Si la geolocalización está activa, entonces se mostrará la opción de desactivarla. En caso contrario solo será visible la opción de activarla.

3.5.5. Pantalla de recomendación

Esta pantalla (Figura 3.10) permite al usuario introducir todos los parámetros para configurar el plan de ocio a solicitar. Entre las opciones disponibles encontramos:

- **Amigos en el plan**. En este apartado se muestra una lista con todos los amigos que hay actualmente en el plan. Por defecto, el sistema de Madrid Live detecta todos amigos los cercanos y los agrega a dicha lista. El usuario puede agregar o eliminar amigos a su gusto.

- **Grupos**. Da acceso al usuario a las opciones de guardar o cargar grupos personalizados. En caso de cargar uno, se eliminarán todos los amigos existentes en el plan y se cargarán todos los pertenecientes a dicho grupo.

- **Horario**. Por defecto Madrid Live elige una franja horaria para el plan que va desde la hora actual hasta cinco horas más adelante. El usuario tiene la opción, también, de elegir un horario manualmente.

- **Actividades**. Aquí el usuario podrá marcar aquellas actividades que le gustaría tener en su plan – o desmarcar las que prefiere no tener –. A

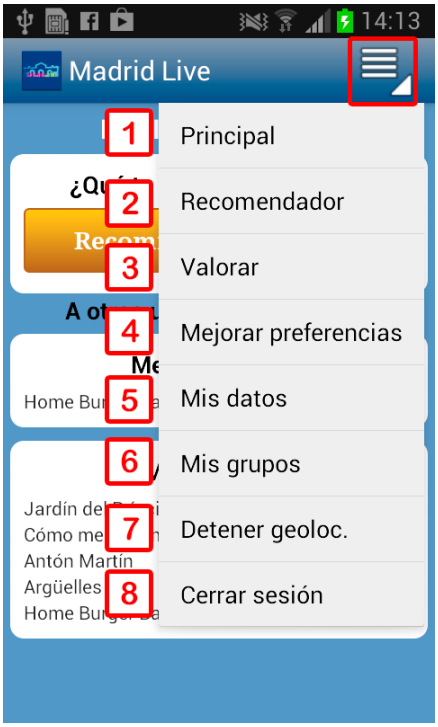


Figura 3.8: Madrid Live: menú principal

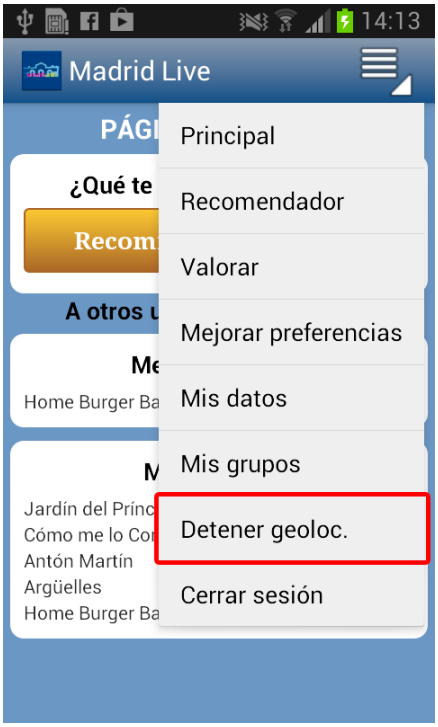


Figura 3.9: Madrid Live: la geolocalización

la hora de la recomendación, Madrid Live puede recortar o modificar dichas actividades si es necesario para que encajen mejor con el perfil del usuario.

- **Lugar**. La posición seleccionada para el plan por defecto es la del usuario solicitante que, como se ve en la Figura 3.10, puede ser cambiada manualmente a través de un sencillo mapa de Google Maps.

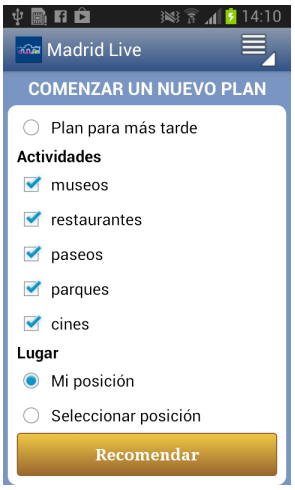
Tras haber configurado el plan a su gusto, el usuario puede hacer click en el botón "Recomendar" para solicitar a Madrid Live un plan de ocio que encaje con estas preferencias.

3.5.6. Resultado de la recomendación

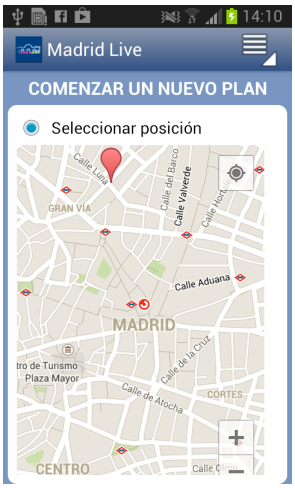
Una vez que el sistema ha devuelto una recomendación de plan de ocio, esta pantalla (Figura 3.11) se encarga de su visualización. La sección de resultado de la recomendación muestra una lista de actividades del plan con toda la información disponible sobre ellas:

- *Icono con el tipo de actividad (cine, restaurante, museo, paseo o parque).*

(a) Amigos en plan y horario



(b) Actividades en plan



(c) Lugar de inicio

Figura 3.10: Madrid Live: sección de recomendación

Figura 3.11: Madrid Live: resultado de una recomendación

- *Nombre y horario de la actividad a realizar.*
- *Dirección.*
- *Imagen del lugar.*
- *Puntos de interés.*
- *Enlace a Google Maps con la posición.*
- *Enlace a la web de la actividad.*

El usuario puede aceptar o rechazar la recomendación ofrecida por Madrid Live. En caso de aceptarla, se agrega a la lista de planes realizados por el usuario y se le da la opción de valorar las diferentes actividades.
Tanto si acepta como rechaza la recomendación, el sistema analiza la respuesta y actualiza el perfil de preferencias del usuario.

3.5.7. Valoración de actividades

Una vez que se ha realizado alguna actividad se mostrará en la pantalla de valoración de actividades (Figura 3.12), donde el usuario puede votar los

lugares ya visitados.

Estas valoraciones sirven para mejorar el perfil de preferencias del usuario así como perfeccionar futuras recomendaciones a otros usuarios.

3.5.8. Datos personales

Esta pantalla (Figura 3.13) muestra datos muy básicos del usuario con la sesión activa. En concreto: nombre completo, dirección de correo, fecha de nacimiento (si está disponible) y sexo (si está disponible).

Figura 3.12: Madrid Live: valoración de actividades

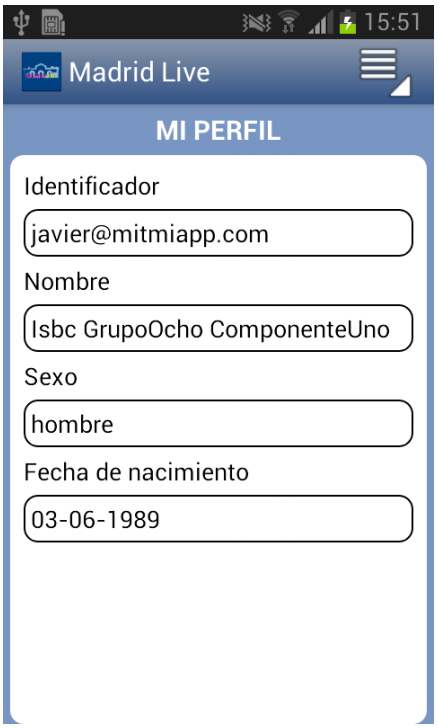
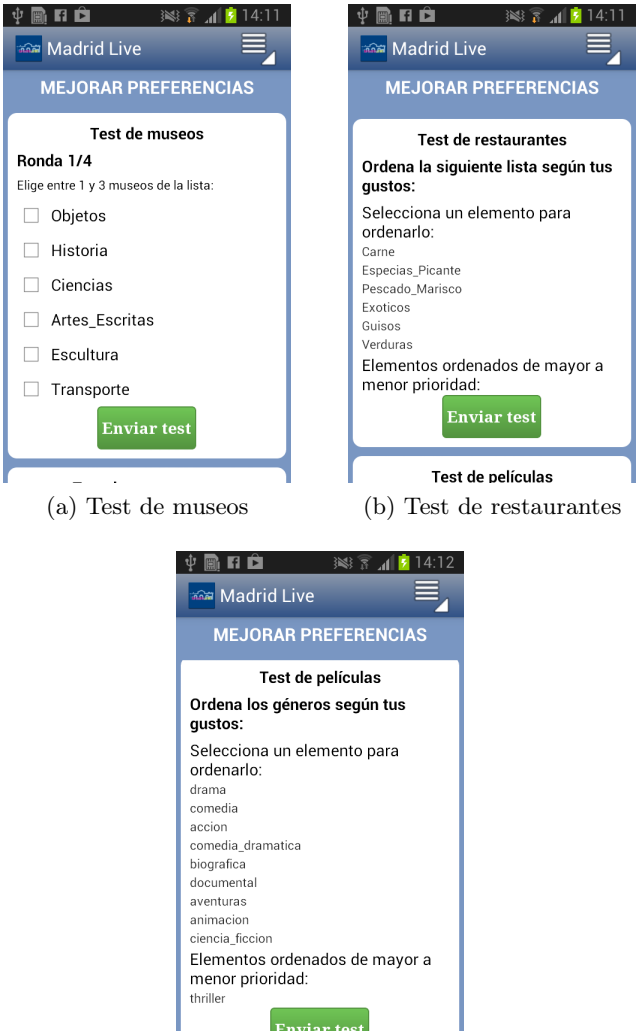


Figura 3.13: Madrid Live: datos personales

3.5.9. Tests de preferencias

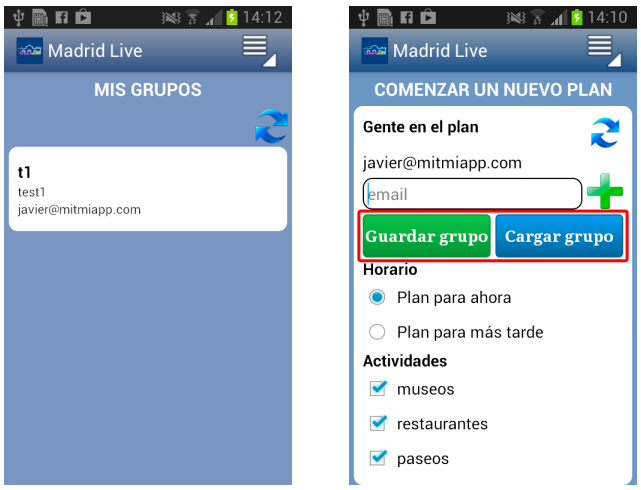
A través de esta pantalla (Figuras 3.14) el usuario puede realizar una serie de tests de preferencias relacionados con museos, restaurantes y cines. En el caso de museos, el usuario debe realizar 4 rondas marcando los tipos de exposiciones que más le atraigan. Para restaurantes y cines bastará con que ordene una lista de tipos de comidas y películas de mayor a menor orden de preferencia.
Estos tests, que ya se han detallado en la Sección 3.3.4, pueden ser repetidos



(a) Test de museos (b) Test de restaurantes

(c) Test de películas

Figura 3.14: Madrid Live: tests de preferencias



(a) Lista de grupos creados

(b) Guardado y carga de grupos

Figura 3.15: Madrid Live: gestión de grupos

libremente y contribuyen a mejorar el perfil de gustos del usuario y afinar en futuras recomendaciones.

3.5.10. Los grupos

Otra de las funcionalidades de Madrid Live es permitir al usuario crear o cargar grupos de amigos personalizados. Es decir, agregar una serie de usuarios y englobarlos bajo un nombre común. Por ejemplo: yo como usuario puedo incorporar a un mismo grupo todos los miembros más cercanos de mi familia y denominarlo "Familia".
Todos los grupos creados se pueden visualizar a través de la pantalla de "Mis grupos" (Figura 3.15), en la que, si pulsamos sobre uno de los grupos mostrados en la lista, se nos dará la opción de eliminarlo.
Tanto el guardado como la carga de grupos se llevan a cabo a través de la pantalla de recomendación.

Conclusiones

A lo largo de este capítulo hemos detallado, por un lado, el funcionamiento general de Madrid Live como sistema de recomendación y la interfaz o *frontend* implementada como aplicación Android.
También hemos expuesto las diferentes fuentes de información usadas por el sistema para obtener datos de las actividades de ocio y su importancia dentro del sistema de recomendación.

Además, hemos hablado sobre la recolección de datos del usuario para el modelado de su perfil de preferencias y su utilidad dentro del sistema.

Todo lo mencionado será de ayuda en el próximo capítulo, donde entraremos en detalles de diseño y funcionamiento mucho más concretos relacionados con las funcionalidades que hemos ido viendo.

Capítulo 4

Arquitectura de Madrid Live

El sistema Madrid Live sigue una estructura cliente - servidor tradicional (Figura 4.1). Los clientes Android (*frontend*) realizan peticiones al *backend* a través de una API o capa de servicios. La parte de *backend* se ha dividido en dos módulos bien diferenciados: API y núcleo.

1. ***Núcleo:*** es el módulo principal de Madrid Live. Engloba todos los componentes básicos necesarios para el funcionamiento del sistema como pueden ser los recomendadores de actividades o el gestor de perfiles de usuario. Además, se encarga también del acceso y gestión de la información proporcionada por la base de datos u otros servicios externos (guiadelocio.com, Google Places, etc.) ya tratados en la Sección 3.2.

2. ***API o capa de servicios:*** es la capa que actúa de intermediaria entre los clientes (aplicaciones Android) y el núcleo del servidor. Está implementada en J2EE y ofrece multitud de llamadas para los diferentes servicios ofrecidos (ver Apéndice B).
   Cuando una petición llega a la API, se envían los datos de la misma al núcleo para su procesamiento. Acto seguido se devuelve la respuesta oportuna al cliente. Por ejemplo: cuando se solicita una recomendación, la petición vendría del cliente y entraría a la API, que solicitaría al núcleo dicha recomendación con los parámetros que el usuario haya introducido. Cuando el núcleo terminase de procesar la petición, delegaría de nuevo en la API, que respondería al cliente.

3. ***Cliente:*** se podría ver como la *interfaz* o *frontend* de Madrid Live. Se trata de una aplicación (nativa) desarrollada en Android y es la única interfaz para que el usuario interaccione con el sistema. Se encarga de recoger las acciones del usuario y mostrarle los resultados de las diferentes operaciones.
   Como se muestra en la Figura 4.1, este cliente Android recibe información de otros servicios adicionales como la red social Facebook (a través

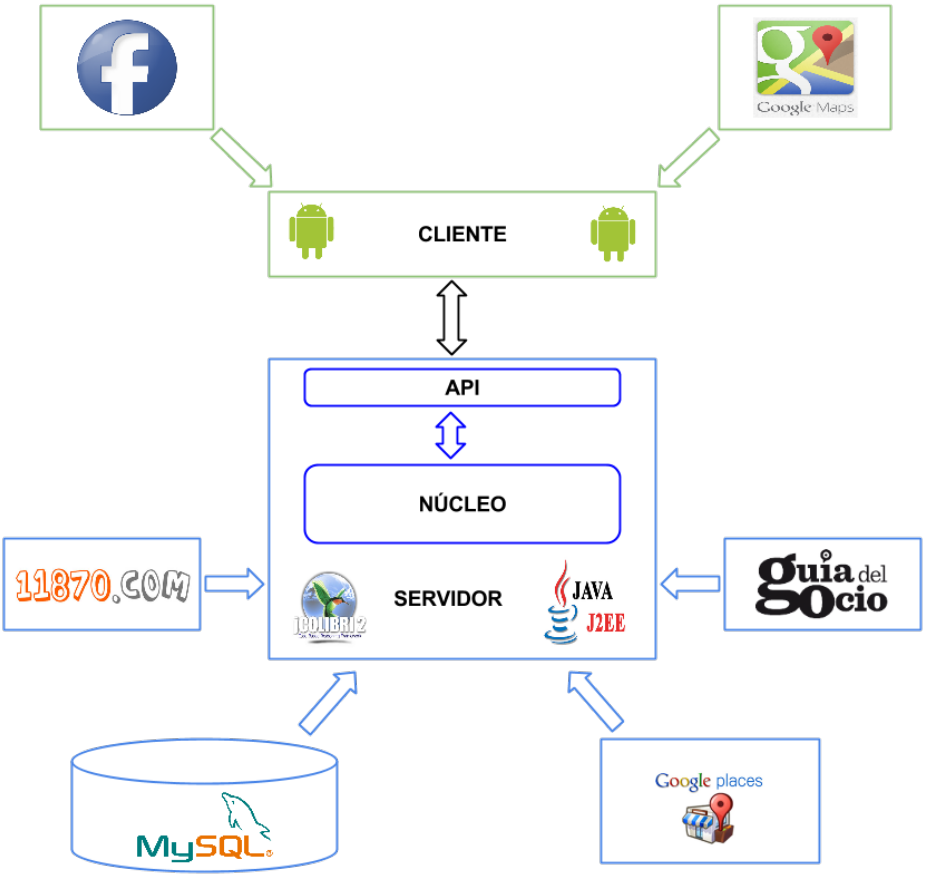


Figura 4.1: Arquitectura de Madrid Live

de la cual el usuario accede a la aplicación) o Google Maps (empleado para ofrecer funcionalidad relacionada con el posicionamiento). Gracias a las ventajas de los dispositivos móviles también nos es posible recopilar gran cantidad de información acerca del contexto del usuario.

4.1. Núcleo del sistema

En esta sección abordaremos todos los detalles concretos de implementación de los componentes más internos de la parte de *backend* como pueden ser la gestión de la base de datos o el sistema de recomendación.

Este núcleo, que está desarrollado completamente en Java y forma parte de un proyecto completamente independiente, se encarga de todo el procesamiento y gestión de datos y contiene todas las funcionalidades que necesitará la capa de servicios (API).

A continuación vamos a profundizar un poco más en los aspectos más relevantes de esta parte del sistema:

4.1.1. Gestión de la base de datos

Madrid Live trabaja con gran cantidad de información relacionada tanto con los usuarios del sistema como con el módulo de recomendación, la mayor parte de la cual se encuentra almacenada en la base de datos MySQL albergada en *backend*.

Para facilitar el acceso a dicha base de datos se ha creado un gestor central que mantiene y refresca las conexiones a MySQL siempre que sea necesario. Se trata de la clase `MySQLConnector.java`, que contiene las credenciales de acceso a MySQL y permite obtener una conexión lista para ser usada a través del método `getConnection()`.

Por otro lado, para organizar el acceso a las diferentes tablas de la base de datos se han creado diversos modelos o gestores de datos[1]. Cada modelo es una clase diferente Java que proporciona métodos de acceso y modificación de algún recurso concreto. Por ejemplo, existe una clase `CinesSQL.java` que engloba toda la gestión de datos relacionada con las películas.
No siempre existe un modelo por cada tabla de la base de datos sino que un mismo modelo puede servir para acceder a varias a la vez. Sin ir más lejos, en el caso anterior, `CinesSQL.java` maneja varias tablas a la vez (cines, sesiones, actores, etc.), con lo que se consigue abstraer la implementación concreta en la base de datos del uso de cada recurso.

[1] Los equivalentes a los usados en arquitecturas MVC

4.1.2. Gestión de datos de la red social

Como se ha comentado en otras secciones anteriores, Madrid Live está integrado con Facebook. Siendo más precisos, el sistema de Madrid Live está preparado para funcionar con los datos de una red social genérica. Nosotros hemos elegido Facebook como red social concreta pero podría realizarse una migración a cualquier otra.

De esta red social, el sistema recoge una serie de información que será de utilidad a la hora de crear un perfil social y de confianza entre los miembros de un grupo. Es importante mencionar que, por defecto, Madrid Live trata de solicitar toda la información posible, pero en el caso de que alguna no esté disponible simplemente se ignoraría y se trabajaría sin ella. Por ejemplo: supongamos que cambiamos la red social a Twitter que, por el motivo que sea, no puede ofrecernos las fotos subidas por un usuario. En este caso los datos correspondientes quedarían vacíos y se ignorarían en cálculos posteriores.

Entre la información recuperada de la red social tenemos:

- **Datos de usuario**: se recogen algunos muy básicos como el identificador único de Facebook, nombre completo, email, fecha de nacimiento y género.
- **Amigos**: almacenamos tan solo la lista de amigos de cada usuario. Es decir, información que nos permita saber quién es amigo de quién.
- **Mensajes**: solamente se recogen de estos los datos que permiten conocer la interacción entre usuarios. Por ejemplo: el identificador de mensaje, la fecha, el emisor y receptor.
- **Fotos**: de nuevo, no se almacena la foto en sí sino solamente la información que permita conocer qué fotos ha subido cada usuario.
- **Etiquetas**: se recogen las etiquetas de usuarios en las fotos anteriores. De esta forma, sabríamos qué personas están etiquetadas en qué fotos subidas por cada usuario.
- **Publicaciones**: en este caso recolectamos información sobre las publicaciones (*posts*) de cada usuario. Se guardan tanto las publicaciones dirigidas a otros usuarios (p.ej.: mensaje en el muro) como las no dirigidas (p.ej.: cambios de estado). Como en los casos anteriores, no nos importa el contenido de la publicación sino la interacción en sí.
- ***Likes***: se recoge información acerca de qué usuarios han hecho un *like* ("me gusta" de Facebook) en contenidos de otros usuarios. Por ejemplo, alguien ha dado un *like* a un cambio de estado de otra persona.

- **Páginas generales**: se guardan de forma independiente los *likes* e interacciones de los usuarios con páginas generales (p.ej.: eventos, películas, etc.).

Toda esta información se almacena en varias tablas de la base de datos cuyo gestor o modelo asociado es `DatosSQL.java`, en el que encontramos multitud de funciones que permiten un sencillo manejo de toda esta información.

4.1.3. Sistema de geolocalización

Desde el punto de vista de servidor, la parte de geolocalización se limita exclusivamente a almacenar la última posición conocida de los usuarios. Será el cliente el que la solicite cuando sea necesario y la trate de forma apropiada. Esta información, de nuevo, se almacena en la base de datos MySQL y tiene un modelo asociado: `UsuarioSQL.java`.

4.1.4. Gestión de las valoraciones

Cuando se valoran actividades, dicha puntuación tiene varios efectos: se actualizan las preferencias del usuario por ese tipo de actividad, se calcula una nueva valoración media para la actividad concreta y se actualiza también la puntuación media de la plantilla de actividades de la que forma parte la valorada. A continuación lo explicaremos en detalle:

4.1.4.1. Actualización de los gustos del usuario

Madrid Live almacena en servidor datos acerca de los gustos de los usuarios sobre los diferentes tipos de museos, películas o restaurantes. Se mantienen varias tablas en las que se guardan todos los tipos de cada una de estas actividades con las preferencias de cada usuario por cada uno de ellos.

Los formatos, valores y actualizaciones de los datos son:

- **Restaurantes**: cada uno de ellos tiene asociado uno o varios tipos de comida (carnes, pescado, guisos, etc.). Como cada persona tiene gustos diferentes respecto a estos tipos de comida, el sistema almacenará las preferencias de cada usuario sobre los mismos.
  Cada tipo de comida tendrá asociado un valor entre 0.0 y 1.0 donde 0 significa que al usuario no le gusta ese tipo de comida y 1 que le gusta mucho.
  Cuando un usuario valore positivamente un restaurante que tenga un determinado tipo de comida asociado el valor de dicho tipo de comida se incrementará en el perfil de usuario. Del mismo modo, se hará lo propio si lo valora de forma negativa.

En concreto, dado que las puntuaciones pueden variar entre 0 y 5, hemos reflejado la correspondiente variación en el valor del perfil del usuario en la Tabla 4.1.
Por ejemplo, si un usuario tiene un peso de 0.7 en el tipo de comida "carne" y valora con un 2 un restaurante cuya especialidad son las carnes, entonces este peso se quedaría en 0.64.

- **Museos**: al igual que en el caso anterior, se han establecido diversos tipos de museos en función de las temáticas de sus exposiciones. Por ejemplo: pintura, arquitectura, fotografía, etc.
  Cada tipo de museo se almacenará, de nuevo, haciendo uso de una valoración entre 0.0 y 1.0 significando 0 que el usuario evita ese tipo de museos y 1 que le gustan mucho.
  Cuando los usuarios voten un museo se actualizarán sus perfiles acorde a la valoración realizada. En caso de dar una puntuación negativa se reducirá la preferencia por dicho tipo de museo y al puntuarlos positivamente se incrementará (Tabla 4.1).

- **Cines**: como ocurre en los dos casos anteriores, de nuevo, cada tipo o género de película tiene asociado una puntuación entre 0 y 1. En este caso, dicha puntuación se actualizará siguiendo el mismo modelo mencionado para restaurantes y museos.

Puntuación	0	1	2	3	4	5
Variación	-0.1	-0.06	-0.02	+0.02	+0.06	+0.1

Tabla 4.1: Variación de los pesos de museos, restaurantes y cines en función de la puntuación del usuario

4.1.4.2. Nueva puntuación para la actividad

Cada una de las actividades que recomienda Madrid Live tiene asociada una puntuación que se corresponde con la valoración media dada por todos los usuarios que han votado esa actividad concreta, lo que constituye una fuente de información muy útil a la hora de recomendar actividades. Por ejemplo, supongamos que el sistema encuentra dos museos que encajan a la perfección con los gustos del usuario. Uno de ellos es el Reina Sofía y otro el Museo del Prado. Como solo debe recomendar uno de ellos, lo siguiente a comprobar serán las valoraciones de otros usuarios para esos tipos de museos. Si suponemos que el Reina Sofía tiene una valoración media de 3 y el Museo del Prado un 5, Madrid Live tratará de recomendar siempre el Museo del Prado frente al Reina Sofía.

Cuando un usuario puntúa una determinada actividad, además de actualizar los pesos en las preferencias de dicho usuario, también se calcula la nueva valoración media para la actividad.

4.1.4.3. Nueva puntuación para la plantilla

Al igual que las puntuaciones medias de actividades individuales son importantes, también lo son las valoraciones medias de plantillas de actividades completas. Por ello, como ocurría en el caso anterior, Madrid Live también almacena puntuaciones medias de los usuarios para determinadas plantillas de ocio. Por ejemplo, supongamos un plan recomendado a un usuario que consta de un paseo, un parque y un cine de 15:00 a 21:00. El usuario ha valorado el paseo (el que fuese) con un 3, el parque con un 5 y el cine con un 4. Madrid Live no sólo actualizaría las puntuaciones medias correspondientes a las actividades sino que calcularía la valoración media de la plantilla completa y la almacenaría. En este caso, sería una puntuación media de 4.

De nuevo, esta información es de gran ayuda a la hora de recomendar plantillas completas a otros usuarios. Si en el ejemplo anterior la valoración media hubiese sido de 1, está claro que el sistema evitaría recomendar dicho plan a otros usuarios siempre que fuese posible.

4.1.5. Procesamiento del perfil de usuario

Como se ha visto en la Sección 3.3, Madrid Live hace uso de multitud de información relacionada con los gustos o preferencias del usuario que se almacena y actualiza siguiendo el modelo que veremos a continuación:

4.1.5.1. Disponibilidad de horario

Aunque los planes de Madrid Live tienen asociado un horario, el problema es que no siempre se puede encontrar un intervalo de horas que corresponda con el solicitado o esperado por el usuario. Por ello, uno de los datos que se almacenan es la predisposición del usuario a cambiar de horario.
Esta disponibilidad al cambio de horario se almacena en el perfil de usuario como un valor que oscila entre 0 y 100. En este caso, 0 significaría que el usuario no tolera en absoluto los cambios de hora del plan y 100 que es totalmente flexible respecto a esto.
Por defecto el sistema asigna el valor medio (50) a todos los nuevos usuarios y lo incrementa o reduce en 10 puntos (un 10 %) cada vez que el usuario acepta o rechaza respectivamente un plan fuera del horario inicial.

4.1.5.2. Predisposición a desplazarse

Otro de los parámetros influyentes en un plan de ocio es la zona donde se realiza. El usuario puede seleccionar libremente dónde le gustaría que estuviesen las actividades a realizar pero, en ocasiones, no es posible encontrar las actividades deseadas cerca de la zona esperada por el usuario, lo que nos ha llevado a guardar un nuevo dato en el perfil de preferencias del usuario: la disponibilidad de desplazamiento.
Al igual que en el caso de los horarios, el valor en el perfil varía entre 0 y 100, significando 0 que el usuario no acepta planes que estén muy lejos de la zona esperada y 100 que a dicho usuario no le importa en absoluto desplazarse si el plan merece la pena.
Como en el caso anterior, por defecto el sistema asigna un valor medio (50) a esta propiedad y lo aumenta o reduce en 10 puntos (un 10 %) cada vez que el usuario acepta o rechaza respectivamente planes que están lejos de la zona prevista.

4.1.5.3. Tipos de actividades

Es importante también para Madrid Live conocer los gustos de los usuarios por cada tipo de actividad a recomendar (museo, cine, parque, paseo o restaurante) ya que resulta de gran ayuda cuando el sistema debe descartar alguna actividad solicitada por el usuario y reemplazarla por otra. Por ejemplo: si el usuario pide un museo a las 7 de la mañana, Madrid Live deberá descartar dicha actividad (por estar fuera del horario) y deberá reemplazarla por otra para ofrecer un plan de ocio para ese rango de horas. En este caso se consultarían los datos de las preferencias del usuario por cada tipo de actividad y se escogería la que más guste al usuario y pueda ser realizada en ese horario.
Para lograrlo, el sistema almacena un campo para cada tipo de actividad dentro del perfil de preferencias del usuario. Cada uno de éstos puede tomar valores entre 0 y 100, significando 0 que el usuario detesta ese tipo de actividad y 100 que le encanta.
El método de actualización de estos valores es muy similar al empleado en horarios y desplazamientos: cuando el usuario se registra, sus preferencias por cada uno de los tipos de actividad se inicializa a un valor medio (50). Cuando el usuario acepta planes en los que se incluye un determinado tipo de actividad, se actualiza su perfil de preferencias incrementando en 10 puntos (un 10 %) sus gustos por ese tipo de actividad. Del mismo modo, cuando el usuario rechaza un plan en el que se incluye un determinado tipo de actividad, se decrementa el valor de preferencia por ese tipo de actividad en 10 puntos.

4.1.5.4. Test de museos

Como se ha visto en la Sección 4.1.4, el servidor almacena en la base de datos el perfil de preferencias de un usuario respecto a los diferentes tipos de exposiciones. Para cada ello, se marca cada tipo de museo con un valor entre 0 y 1, siendo 0 la puntuación más baja y 1 la más alta.
Durante la realización del test de museos (ver Sección 3.3) el usuario pasa por 4 rondas, en cada una de las cuales se recogen los tipos de museos seleccionados por el usuario como sus preferidos. Cuando se han terminado todas las rondas, lo que se obtiene es una lista ordenada de los diferentes tipos de exposiciones por orden de preferencia para el usuario.
Es en base a esta lista como el servidor asignará los nuevos pesos dentro del perfil de preferencias del usuario, dando mayores puntuaciones a los tipos de exposición que más agradan al usuario y menores al resto.
Además, el algoritmo ELO se encarga de que las diferencias entre los valores fijados sean lo más equilibradas posibles.

4.1.5.5. Test de restaurantes

Al igual que en el caso de los museos, también se almacenan datos en el perfil de preferencias del usuario relacionados con los diferentes tipos de restaurantes. De nuevo, cada uno de ellos recibe una puntuación entre 0 y 1 para cada usuario. El valor 0 indicaría que al usuario no le agrada nada un determinado tipo de restaurante y 1 que lo adora.
En este caso, tras la realización del test de restaurantes (ver Sección 3.3) llega al servidor una lista ordenada que incluye los diferentes tipos de comidas con los elementos ordenados de mayor a menor preferencia para el usuario.
A diferencia de lo que ocurría con los museos, en este caso los valores concretos dentro del perfil de preferencias se asignan en función de la posición que ocupa un determinado tipo de restaurante en la lista ordenada. En concreto, estos valores se recogen en la Tabla 4.2.

Posición	1º	2º	3º	4º	5º	6º
Nuevo peso	1.0	0.8	0.6	0.4	0.2	0.0

Tabla 4.2: Variación de los pesos de restaurantes tras el test de preferencias

4.1.5.6. Test de cines

Por último, en el caso de los cines, también se guardan datos en el perfil de preferencias del usuario relacionados con los diferentes tipos de películas. Como en los dos casos anteriores, cada uno de ellos tiene fijada una valoración entre 0 y 1, donde 0 significa que ese género de película no es del agrado

del usuario y 1 que le encanta.
Tras la realización del test de cines (ver Sección 3.3) el servidor recibe una lista ordenada con todos los tipos de películas. Los elementos llegan ordenados de mayor a menor preferencia para el usuario.
Como en el caso de los restaurantes, los nuevos pesos asignados al perfil de preferencias de películas del usuario se asignan en función de la posición de cada género cinematográfico en la lista ordenada resultante. En este caso, como hay gran cantidad de géneros (más de 20), solamente los 10 primeros recibirán una puntuación positiva. Los restantes mantendrán una puntuación de 0. Todo esto se ha reflejado en la Tabla 4.3.

Posición	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	Resto
Nuevo peso	1.0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1	0.0

Tabla 4.3: Variación de los pesos de cines tras el test de preferencias

4.1.6. Módulos de recomendación

El sistema de recomendación de Madrid Live está compuesto por varios módulos recomendadores que, una vez unidos, conforman la funcionalidad de recomendación de actividades completa.

Para simplificar la comprensión empezaremos explicando brevemente cada uno de estos pequeños módulos y su acoplamiento con el resto. Comenzaremos por los sistemas recomendadores más específicos e iremos avanzando hacia los más generales, explicando, para finalizar, la unión de todos ellos en los módulos más genéricos.

Como anotación general, decir que todos los módulos recomendadores están basados en un proceso CBR[2] y la implementación concreta se ha llevado a cabo en todos los casos haciendo uso del framework JColibrí[3].

4.1.6.1. Recomendador de museos

En primer lugar nos encontramos el recomendador de museos, que busca en la base de datos un posible museo a recomendar al usuario. Toda la implementación de este módulo de recomendación se encuentra en la clase `RecomendadorMuseos.java`.

El proceso que sigue este recomendador es:

[2] http://es.wikipedia.org/wiki/Razonamiento_basado_en_casos
[3] http://gaia.fdi.ucm.es/research/colibri/jcolibri

- **Obtención de los datos**.
  En primer lugar, el sistema recupera todos los posibles museos a recomendar de la base de datos y mapea cada uno de ellos en un objeto independiente (`MuseoDescription.java`) que incluye todos los datos necesarios en los que se basará la recomendación (tipo de exposición, localización, horario, etc.).
  A continuación, con esta base de casos recuperada, se asignan una serie de pesos a cada uno de los atributos que se tendrán en cuenta para evaluar cada museo. Este peso indicará la "importancia" de dicho atributo frente al resto. Por ejemplo, si se informa a este recomendador de que el usuario prefiere planes cercanos a la posición prevista, lo que hace el sistema es incrementar el peso del atributo "posición" para que éste cobre importancia frente al resto.
  Tanto en este recomendador concreto como en los demás, jugar con los valores de los pesos asignados a cada atributo permite afinar bastante las recomendaciones ofrecidas en función de las preferencias del usuario concreto.

- **Ciclo CBR**.
  Con la base de casos ya cargada y los pesos de los atributos fijados, se ejecuta el ciclo CBR y se obtiene así una lista ordenada de cada uno de los museos junto con un "valor de satisfacción" que indicará cómo se ajusta dicho museo a las preferencias del usuario. Cuanto mayor sea el valor, más adecuada sería la recomendación de ese elemento.
  En principio se recuperan todos los museos ordenados por valor de "satisfacción" de mayor a menor (para facilitar la gestión posterior).

- **Filtrado y recuperación de las soluciones**.
  Una vez que se han analizado todos los museos y se les han asignado sus correspondientes valoraciones, se procede a aplicar un doble filtrado. Se recorre la lista de los museos en orden descendente de valor de "satisfacción" (para tratar de escoger siempre el mejor de ellos).
  Para cada museo, se comprueba, en primer lugar, si cumple las restricciones horarias necesarias. De no hacerlo, se descarta y se procede con el siguiente.
  A continuación se comprueba que el usuario para el que se está recomendando no haya visitado ese museo recientemente. De haberlo hecho, se descartaría esa opción.
  Se devuelve como recomendación el museo con el mayor valor de "satisfacción" que cumpla los dos filtros anteriores mapeado haciendo uso de la clase `MuseoSolution.java`.

- **Función de agregación**.
  Si la recomendación realizada es para un grupo en vez de un usuario individual, el procedimiento seguido es muy similar. La principal di-

ferencia es que se solicita un museo a recomendar para cada usuario del grupo, obteniendo así, tras todas las recomendaciones individuales, una lista de museos que se filtrará haciendo uso de una función de agregación (se detallará en otro apartado de esta sección) que analizará todos los elementos recuperados y elegirá el que mejor satisfaga al grupo.

4.1.6.2. Recomendador de parques

El siguiente módulo del que hablaremos es el recomendador de parques, que localiza en la base de datos un posible parque a recomendar al usuario. Este recomendador es más simple que el de museos ya que la información con la que se trabaja para los parques es mucho menor. De hecho, se juega sobre todo con las valoraciones de otros usuarios y la localización del parque. La implementación concreta de este módulo se encuentra en la clase `RecomendadorParques.java`.

El recomendador realiza el siguiente proceso:

- **Obtención de los datos**.
  El primer paso realizado por el recomendador antes del ciclo CBR es la búsqueda en la base de datos de todos los posibles parques a recomendar y el mapeo de cada uno de los elementos recuperados en un objeto de la clase `ParqueDescription.java`, que engloba todos los atributos de los que se hará uso en la recomendación.
  Una vez que esta base de casos ha sido generada, se procede a asignar pesos a los diferentes atributos a evaluar para cada parque (localización, valoraciones de otros usuarios, etc.), con lo que conseguiremos dar mayor o menor importancia a una u otra propiedad de los parques. En este caso, a diferencia de los museos, el número de atributos es mucho menor dada la naturaleza de los parques.

- **Ciclo CBR**.
  Una vez que se han recuperado correctamente todos los posibles parques, se han almacenado en una base de casos y se ha asignado un peso a cada atributo, se procede a ejecutar el ciclo CBR, que evaluará cada uno de los elementos de la base de casos y los devolverá en una lista ordenada junto con un "valor de satisfacción", que indicará el grado de ajuste de dicho parque a las preferencias del usuario. A mayor valor, mejor ajuste.
  Aunque se podrían recuperar solamente los N elementos de mayor "valor de satisfacción", se trabaja con toda la lista completa para aplicar posteriores filtrados.

- **Filtrado y recuperación de las soluciones**.

Con la lista de parques ya evaluada y ordenada de mayor a menor valor de "satisfacción" se aplica un único filtrado sobre el horario del parque para confirmar que resulta apropiado para las restricciones impuestas. En caso de no ser así, se descartaría la actividad y se continuaría buscando otro posible parque dentro de la lista.
Tras el filtrado se devuelve como resultado aquel parque que tenga el mayor valor de "satisfacción" y que cumpla el filtro anterior, mapeado en un objeto de la clase `ParqueSolution.java`.

- **Función de agregación**.
  A diferencia de otros módulos de recomendación, cuando se solicita una recomendación grupal en vez de individual, no se solicita un parque para cada uno de los usuarios y se aplica posteriormente una función de agregación. El motivo es que no existe ningún tipo de preferencia específica para un "tipo" de parque almacenada en los perfiles de preferencias de los usuarios.
  Como se juega principalmente con las valoraciones de otros usuarios y la posición del parque, da igual para qué usuario estemos recomendando, siempre trataremos de ofrecer el más cercano y con mejores puntuaciones.
  Sí se tiene en cuenta, no obstante, la predisposición media del grupo a desplazarse para ir a dicho parque.

4.1.6.3. Recomendador de paseos

Un módulo muy similar al anterior (parques) es el recomendador de paseos, que localiza en la base de datos un posible paseo a recomendar al usuario. Al igual que en el caso de los parques, este recomendador es más simple que el de museos ya que se trabaja con menor cantidad de información (valoraciones de otros usuarios y la localización del paseo).
La implementación concreta de este módulo se encuentra en la clase `RecomendadorPaseos.java`.

Para la recomendación de un paseo, este módulo realiza los siguientes pasos:

- **Obtención de los datos**.
  Como primera acción, el recomendador de paseos accede a la base de datos local de servidor y genera una base de casos que contiene todos los posibles paseos a recomendar, cada uno de ellos mapeado en un objeto de la clase `PaseoDescription.java`, en el cual se recogen todos los atributos relevantes para realizar la recomendación.
  Tras la obtención de la base de casos completa, el siguiente paso realizado es la ponderación de los diferentes atributos característicos de cada paseo (localización, monumentos, etc.) asignando un peso a cada

uno de ellos. A mayor peso, mayor importancia tendrá dicha propiedad dentro de la recomendación general, lo que permite realizar ajustes en función de las preferencias del usuario. Por ejemplo: si estamos ante uno al que no le gustan los desplazamientos, el atributo de "localización" del paseo tendrá un mayor peso frente al resto.

- **Ciclo CBR**.
  El siguiente paso realizado tras la carga de la base de casos y la asignación de los pesos a los atributos es la ejecución del ciclo CBR para la evaluación de cada uno de los posibles paseos. Al igual que en los demás recomendadores, esta evaluación se lleva a cabo asignando un "valor de satisfacción" a cada uno de los posibles paseos.
  Una vez conseguida esta lista ordenada de paseos, se trabaja con ella completa para aplicar un filtrado.

- **Filtrado y recuperación de las soluciones**.
  Para cada paseo de la lista anterior se aplica un filtro bastante simple consistente en una comprobación sobre el horario y la duración del paseo. En caso de que no se ajusten a lo buscado, se descartaría la actividad o se analizaría la posibilidad de recortar algún punto de la ruta del paseo para reducir la duración del mismo y encajarlo en el plan de ocio.
  Como recomendación, se devuelve aquel paseo que cumpla el filtro anterior y tenga el mayor valor de "satisfacción" posible mapeado en un objeto de la clase `PaseoSolution.java`.

- **Función de agregación**.
  Si estamos ante una recomendación grupal, al igual que ocurría en el caso de los parques, solamente se solicita un paseo a recomendar y no se aplica ninguna función de agregación.
  El motivo es que actualmente no existe ningún tipo de elemento distintivo para los diferentes paseos almacenado en las preferencias de los usuarios. Por ejemplo: se podría haber recogido información acerca de los tipos de monumentos o puntos de interés a visitar que, seguramente, tendrán propiedades características que encajarían mejor o peor con cada usuario. En este caso sí habría que solicitar un paseo para cada miembro del grupo y aplicar la función de agregación.
  No obstante, actualmente en la recomendación de paseos se juega con atributos más básicos como la localización o las valoraciones de otros usuarios, que son elementos comunes a los diferentes miembros del grupo.
  A pesar de ello, sí se analizan otros aspectos relevantes para el grupo como la predisposición media del grupo a desplazarse para llegar a la zona de la actividad.

4.1.6.4. Recomendador de restaurantes

Este recomendador, muy similar al de museos, busca en la base de datos un posible restaurante a recomendar al usuario. Toda la implementación de este módulo se encuentra en la clase `RecomendadorRestaurantesCBR.java`.

Para la elección de un restaurante, este recomendador realiza los siguientes pasos:

- **Obtención de los datos**.
  Cada uno de los restaurantes recuperados de la API de 11870.com se encuentra almacenado en la base de datos local, por lo que, lo primero que hace este recomendador es acceder a la misma para generar una base de casos mapeando cada uno de los posibles restaurantes en un objeto (`RestauranteDescription.java`) que incluya todos los atributos relevantes para la recomendación.
  A continuación, de forma común a todos los restaurantes, se asigna un peso que ponderará cada uno de los atributos que intervendrán en el proceso de evaluación (tipo de comida, localización, etc.). Estos pesos no se asignan siempre de la misma forma sino que varían en función del usuario concreto al que se esté recomendando (teniendo en cuenta su perfil de preferencias).

- **Ciclo CBR**.
  Una vez que el sistema ha recuperado todos los restaurantes, los ha almacenado en la base de casos y se han establecido pesos para todos los atributos relevantes, se procede a ejecutar el ciclo CBR, tras el cual se obtendrá una evaluación para cada uno de los posibles restaurantes en función del ajuste a las preferencias del usuario (de nuevo, usando un "valor de satisfacción").
  Esta lista de restaurantes evaluados se pasa completa y ordenada de mayor a menor "valor de satisfacción" a una siguiente fase de filtrado.

- **Filtrado y recuperación de las soluciones**.
  En el caso de restaurantes se aplica un doble filtrado a cada uno de los elementos de la lista anterior recorrida en orden descendente de "valor de satisfacción" (para tratar de elegir siempre el mejor de ellos).
  El primer filtro consiste en comprobar si el restaurante evaluado tiene un horario que permita agregarlo al plan teniendo en cuenta las restricciones impuestas. De no ser así, se descarta y se procede con el siguiente.
  Como segundo filtrado, y para dar mayor variedad a las recomendaciones del sistema, se comprueba que el usuario no haya ido a ese restaurante recientemente. En caso de haberlo hecho, se descartaría también esa opción.

La recomendación ofrecida por Madrid Live será el restaurante que tenga mayor "valor de satisfacción" y que cumpla ambos filtros mapeado haciendo uso de la clase `RestauranteSolution.java`.

- **Función de agregación**.
  Solo en el caso de que la recomendación haya sido solicitada para un grupo, se recupera un posible restaurante para cada uno de los miembros del mismo. Acto seguido se aplica una función de agregación que evaluará todos los restaurantes recomendados y devolverá como resultado aquel que mejor satisfaga al grupo.

4.1.6.5. Recomendador de cines

Este es el último recomendador de actividades concretas y tiene gran similitud con el de museos y restaurantes. En este caso el recomendador busca en la base de datos una posible película y un cine a recomendar al usuario. Toda la implementación de este módulo de recomendación se encuentra en la clase `RecomendadorCines.java`.

El proceso que sigue este recomendador es:

- **Obtención de los datos**.
  Con ayuda del gestor de la base de datos, el módulo recupera una lista de todas las sesiones de películas del día actual y mapea cada una de ellas en un objeto independiente de la clase (`CineDescription.java`) que recoge los atributos de cada película, cine y sesión a tener en cuenta a la hora de la recomendación (género de la película, localización del cine, horario, etc.). La lista de todos estos elementos compondrá la base de casos a usar en el ciclo CBR.
  El siguiente paso es la asignación de pesos a cada uno de los atributos que se han recogido anteriormente. Como de costumbre, a mayor peso, mayor importancia tendrá dicha propiedad frente a las demás.

- **Ciclo CBR**.
  Tras la creación de la base de casos y la asignación de pesos a los atributos, se lleva a cabo el ciclo CBR para obtener una lista ordenada con las diferentes sesiones de películas junto con sus correspondientes "valores de satisfacción".
  Como en los casos anteriores, a partir de aquí se trabaja con la lista completa de elementos evaluados ordenada de mayor a menor "valor de satisfacción".

- **Filtrado y recuperación de las soluciones**.
  En el caso de las películas y cines, antes de determinar cuál de ellos se recomendará, se lleva a cabo un doble filtrado.

Por un lado, para cada sesión de película se comprueba si encaja con las restricciones de horario impuestas originalmente. En caso de no ser así, dicha sesión es descartada.
Por otro lado, en un segundo filtro, se comprueba que el usuario no haya visto dicha película recientemente. Este filtro cobra mayor importancia en este recomendador que en el de restaurantes, ya que podemos permitirnos enviar a un usuario dos veces al mismo sitio a comer pero sería ilógico recomendarle dos veces la misma película.
Como recomendación, se devuelve aquella película y sesión que mejor encaje con las preferencias del usuario y las restricciones impuestas y que cumpla los dos filtros mencionados. Dicha solución se mapea haciendo uso de la clase `CineSolution.java`.

- **Función de agregación**.
  Cuando la recomendación de un cine se realiza para un grupo y no para un usuario individual, el procedimiento seguido es muy similar, salvo que, en este caso, se solicita una película para cada miembro del grupo. Tras todas las recomendaciones individuales se obtiene una lista de películas que se ajustan a cada uno de los componentes del grupo, que será filtrada haciendo uso de una función de agregación con el objetivo de recuperar la película que mejor encaje con el grupo.

4.1.6.6. Recomendador de plantillas concretas

El recomendador de plantillas concretas es el encargado de tratar de localizar una plantilla de actividades concretas dentro de las ya realizadas por otros usuarios mediante un sistema CBR.

Este recomendador realiza la siguiente secuencia:

- **Obtención de los datos**.
  En primer lugar se recuperan todas las plantillas de actividades realizadas por otros usuarios y se mapean en objetos de la clase `PlantillaDescription.java`, que incluyen todos los atributos necesarios en los que se basará la recomendación (lista de actividades, localización inicial, horario del plan, etc.). La lista de todos estos elementos conformará la base de casos a usar en la fase del ciclo CBR.
  Se recuperan, además, las restricciones impuestas por el usuario para el plan buscado y se mapean en otro objeto de la clase `PlantillaDescription.java` que servirá también de entrada al ciclo CBR.
  Acto seguido se procede a asignar pesos a cada uno de los atributos relevantes a la hora de evaluar cada plan concreto de la base de casos.

- **Ciclo CBR**.

El ciclo CBR recibe como entrada la base de casos previamente cargada, los pesos de los atributos y las restricciones impuestas y ofrece como salida una lista ordenada que contiene todos los planes evaluados con un "valor de satisfacción" que indica cómo de bien se ajusta dicho plan a las preferencias impuestas (a mayor valor, mejor ajuste).
Esta lista ordenada se recupera completa y se pasa a una siguiente fase de filtrado, donde se descartarán todos aquellos planes que no resulten apropiados.

- **Filtrado y recuperación de las soluciones**.
  Antes de devolver un plan de ocio como recomendación, se analiza cada uno de los resultados del ciclo CBR y se aplica sobre él un triple filtrado.
  Por un lado se comprueba si cumple las restricciones básicas impuestas por el usuario (horario, localización, etc.) y se descarta en caso de no ser así.
  A continuación se comprueba que el usuario para el que se ha solicitado la recomendación no haya realizado justo ese plan recientemente (en el último mes).
  Por último se evalúa el nivel o "valor de satisfacción" calculado tras el ciclo CBR, y si éste es menor a 0.75, se deshecha la plantilla evaluada ya que se considera más apropiada la búsqueda de un plan nuevo para mejorar la satisfacción del usuario.
  Se devuelve como recomendación de plan de ocio aquel con el mayor "valor de satisfacción" que cumpla los tres filtros anteriores mapeado haciendo uso de la clase `PlantillaSolution.java`.
  En caso de que ninguna plantilla cumpla los requisitos buscados, se llama al recomendador de plantillas abstractas (que veremos a continuación) para solicitar un conjunto de actividades completamente nuevo.

- **Función de agregación**.
  Si la recomendación realizada es para un grupo en vez de un usuario individual, el procedimiento seguido difiere en que se solicita una plantilla concreta de actividades para cada usuario del grupo, obteniendo así, tras todas las recomendaciones individuales, una lista de planes de ocio que se ajusten a cada uno de los componentes del grupo.
  Tras esto se realiza un filtrado mediante una función de agregación que analizará todos los elementos recuperados y elegirá el que mejor satisfaga al grupo (teniendo en cuenta los filtrados ya mencionados).

4.1.6.7. Recomendador de plantillas abstractas

El recomendador de plantillas abstractas es el encargado de generar una plantilla de actividades completamente nueva cuyos huecos serán rellena-

dos posteriormente con los recomendadores de actividades concretas (cines, restaurantes, parques, paseos y museos). Al igual que en todos los casos anteriores, nos encontramos ante un sistema de recomendación CBR.

En este caso la secuencia de operaciones de este recomendador viene determinada por los siguientes puntos:

- **Obtención de los datos**.
  El primer paso realizado por este recomendador es la recuperación de todas las posibles configuraciones de actividades disponibles y el mapeo en objetos de la clase `PlantillaDescription.java`.
  Se recuperan combinaciones de la forma: 09:00-11:00 paseo, 09:45-15:45 paseo + parque + restaurante, etc. y para cada una de ellas se guardan solamente los datos necesarios en los que se basará la recomendación (lista de actividades, horario del plan, etc.).
  Todas estas posibles combinaciones se almacenan en una lista que será la base de casos empleada posteriormente en el ciclo CBR.
  A continuación se asignan una serie de pesos a cada uno de los atributos que se tendrán en cuenta para evaluar cada plan abstracto, lo que permitirá agregar un valor de "importancia" de unas propiedades frente a otras.
  En principio los atributos más importantes a evaluar serán el horario del plan concreto y la lista de actividades para que encajen lo mejor posible con las restricciones impuestas por el usuario y su perfil de preferencias.

- **Ciclo CBR**.
  Partiendo ya de una serie de restricciones, una base de casos completa y unos pesos fijados para los atributos relevantes, se procede a ejecutar el ciclo CBR para obtener así una lista ordenada de cada uno de los planes abstractos con un "valor de satisfacción" que indicará cómo se ajusta dicho plan a las preferencias del usuario. Cuanto mayor sea este indicador de satisfacción, más adecuada sería la recomendación de ese plan para dicho usuario.
  Con el fin de tener la mayor cantidad de datos posibles, se envía la lista completa de planes abstractos evaluados a un siguiente paso para la recuperación de una solución.

- **Filtrado y recuperación de la solución**.
  A diferencia de otros recomendadores vistos previamente, en este caso no es necesario realizar ningún tipo de filtrado especial sino que se devuelve siempre el plan con el mayor "valor de satisfacción" posible.
  El motivo es que a este recomendador solamente se accede si no hay otra forma de obtener un plan mejor (es decir, es el último recurso) y debe devolver siempre un plan válido.

Se deja, no obstante, esta fase de filtrado preparada y abierta para posibles mejoras.

- **Rellenado de la plantilla**.
  Con el molde de actividades (plan abstracto) ya definido, el recomendador de plantillas abstractas procede a rellenar cada uno de los huecos disponibles con actividades concretas. Para ello hace uso de los diferentes recomendadores de actividades concretas (museos, cines, etc.) según vayan siendo necesarios.
  El sistema de recomendación de plantillas abstractas no hace uso de ningún tipo de función de agregación ya que, de ser necesario, ésta se aplicaría dentro de cada recomendador individual.

4.2. Capa de servicios

La capa de servicios creada para Madrid Live supone una gran ventaja ya que abstrae las funcionalidades más básicas del sistema del cliente que se conecta al mismo. En la implementación actual se ha optado por el sistema operativo y tecnologías de Android para el desarrollo de la parte de cliente, pero, gracias a esta capa de servicios o API, el cliente podría ser migrado fácilmente a plataformas web o entornos iOS. Incluso se podrían tener clientes conectados desde diferentes plataformas.

Esta capa de servicios ha sido implementada completamente haciendo uso de J2EE, cuyo núcleo se reside en los *Servlets*, que son clases Java que se encargan de recibir y devolver las peticiones del cliente. Cada uno de estos *Servlet* se encarga de alguna tarea específica. Por ejemplo: `ValorarServlet.java` tiene como principal función recoger las acciones de valoraciones de actividades realizadas. Puede encontrarse una lista detallada de todos los servicios disponibles en la API en el Apéndice B.

Una vez que un *Servlet* recibe una petición del cliente, se encarga de analizar los datos provenientes del mismo y, en caso de que se trate de una solicitud válida, la envía al núcleo del sistema para su procesamiento.

Cuando dicha petición ha sido procesada por completo, el núcleo envía de nuevo la información necesaria al *Servlet*, que la formatea y entrega al cliente como respuesta a su solicitud.

Para las respuestas a cliente se ha elegido el formato JSON[4] por tratarse de uno de los más comunes y flexibles a la vez que mantiene una gran simplicidad estructural. Hacer uso de un buen formato de salida es muy importante

[4] http://es.wikipedia.org/wiki/JSON

sobre todo al trabajar con tecnologías móviles ya que cuanto más simples y breves sean los datos a enviar, menor tiempo de transimisión habrá y menor consumo de datos del dispositivo u otros recursos como la batería.

4.3. Cliente Android

En la Sección 3.5 ya se detallaron las diferentes pantallas del cliente Android y sus finalidades, por lo que en ésta se tratarán temas de diseño y funcionamiento más concretos como puede ser la gestión de la base de datos local o la conexión con la red social.

No se explicarán todos los módulos del cliente sino solamente aquellos que realmente requieran un mínimo de detalle para su comprensión. Por ejemplo: no se hablará de las conexiones con la capa de servicios del servidor ya que se trata de algo muy común en la tecnología Android.

4.3.1. Actividades

En primer lugar, decir que la aplicación Android se compone de una serie de actividades[5], cada una de las cuales proporciona una funcionalidad diferente al usuario. Por ejemplo: tendremos una actividad para el inicio de sesión, otra para la recogida de valoraciones de actividades, etc. En general, existe una actividad diferente para cada pantalla del sistema (ver Sección 3.5) y pueden encontrarse todas en el paquete `com.ssii.madridlive` del proyecto de cliente.

Como hay determinados comportamientos comunes a todas las actividades, se ha generado una clase principal de la que todas las actividades deben heredar. Se trata de `GeneralActivity.java`, en la que podemos encontrar funcionalidades como el menú principal (se asocia para todas las pantallas), la comprobación de sesión activa en la red social o el comportamiento al pulsar el botón de "volver atrás".

4.3.2. Base de datos local

Al igual que se hacía en servidor, el cliente Android de Madrid Live también mantiene una pequeña base de datos (SQLite[6]) local que contiene aquella información necesaria para el uso de la aplicación.
No se guardan todos los datos como se hace en servidor sino solamente aquellos que se van recuperando durante el uso del sistema y que son susceptibles de ser consultados con frecuencia, lo que supone tanto un ahorro de recursos

[5] http://developer.android.com/guide/components/activities.html
[6] http://es.wikipedia.org/wiki/SQLite

como una mejora de rendimiento ya que permite al cliente obtener datos de forma muy rápida sin tener que establecer una conexión con la capa de servicios.

Los principales datos almacenados de esta forma son:

1. Información básica del usuario (nombre, email, fecha de nacimiento, etc.).
2. Lista de amigos y datos básicos sobre los mismos.
3. Lista de grupos creados y los miembros pertenecientes a los mismos.

En lo que a la conexión con el sistema gestor de la base de datos se refiere, se ha creado una clase `GestorBaseDatos.java` que se encarga de conseguir conexiones listas para ser usadas así como de la creación, actualización y borrado de las correspondientes tablas en función de la versión actual de la base de datos.

Por último, para el acceso a los datos se han creado una serie de modelos o *Data Access Objects* que se encargan de ofrecer las funcionalidades necesarias para el manejo de la información de la base de datos local abstrayendo así al cliente de la implementación y estructuración concreta de estos datos. Tenemos, por lo tanto, un modelo para la gestión de información del usuario (`UsuarioDAO.java`), otro para el manejo de la lista de grupos (`GruposDAO.java`) y otro para la gestión de amigos (`AmigosDAO.java`).

4.3.3. Conexión con la red social

Como ya se ha visto varias veces a lo largo de esta memoria, el sistema de Madrid Live requiere de la integración con un servicio externo que le permita recuperar datos acerca de los usuarios y sus círculos sociales. Por lo general, lo más apropiado para esto es realizar la integración con una red social. En nuestro caso hemos optado por Facebook pero se ha creado una arquitectura para Madrid Live que le permitiría ser conectado a cualquier otro servicio o red social.

Esta abstracción del servicio concreto usado sigue la estructura mostrada en la Figura 4.2 y se compone de las siguientes clases:

- `ConectorRedSocial.java`: se trata de una interfaz que contiene los métodos genéricos que puede usar el cliente para trabajar con la red social o servicio empleado.
  En caso de tener que hacer un cambio, bastará con que la clase que herede de esta interfaz y que esté asignada a la nueva red social redefina

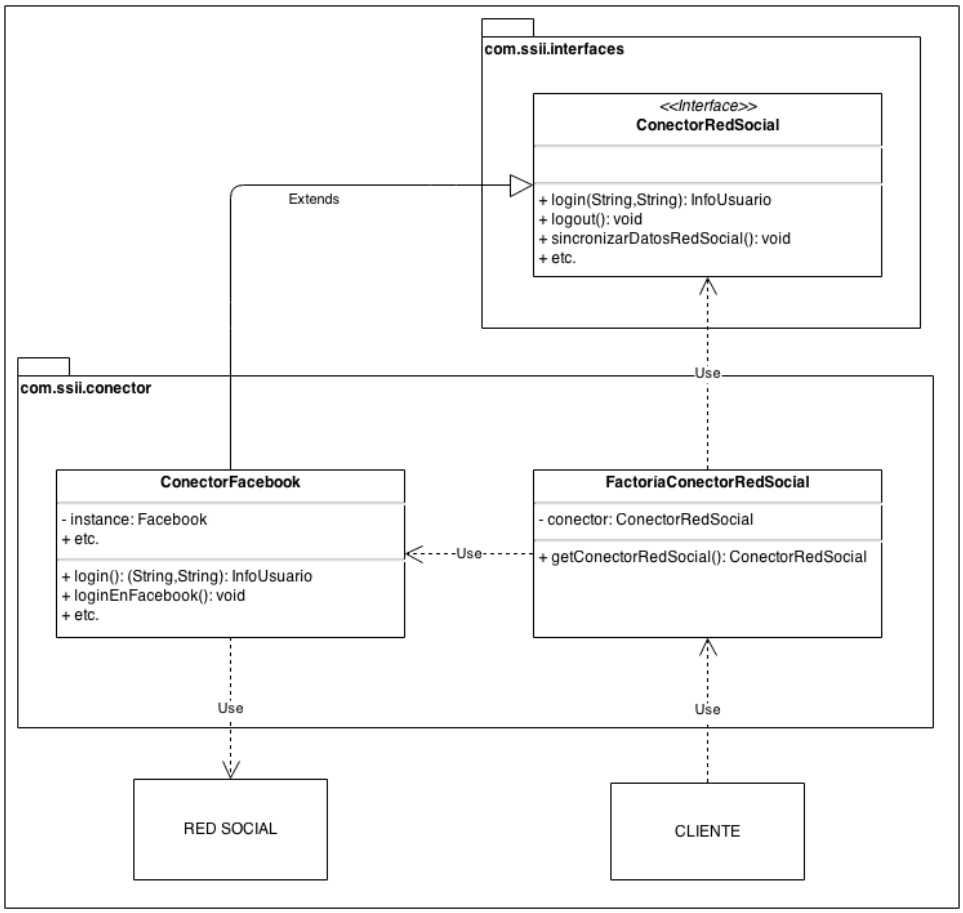


Figura 4.2: Esquema de abstracción de la red social de cliente

correctamente los métodos necesarios haciendo uso de los servicios de la misma.
Algunos de los métodos definidos en esta interfaz son `login()` o `sincronizarDatosRedSocial()`.

- `ConectorFacebook.java`: se trata de una clase que implementa la interfaz anterior y que contiene la redefinición de los métodos de dicha interfaz usando los servicios de Facebook.
  En el caso de Madrid Live, por sencillez de uso, se ha integrado el SDK que proporciona Facebook para la incorporación de los servicios de esta red social en aplicaciones móviles.
  Por ello, por ejemplo, la implementacón de la función `login()` antes mencionada pasaría simplemente por realizar una llamada al método de inicio de sesión de dicho SDK, que se encargará de mostrar el formulario oportuno y recoger y almacenar las credenciales de sesión.
  Es en este punto donde entraría el cambio de red social en caso de ser necesario. Supongamos que ahora deshechamos Facebook y pasamos a Twitter. Bastaría con crear una nueva clase (`ConectorTwitter.java` por ejemplo) que implementase la interfaz `ConectorRedSocial.java` y completase los métodos necesarios haciendo uso, esta vez, de los servicios de Twitter.
  Dado que el cliente realmente manejará objetos de tipo `ConectorRedSocial.java`, no le importa que la implementación concreta esté realizada con uno u otro servicio.

- `FactoriaConectorRedSocial.java`: es una clase factoría[7] que se encarga de mantener la única instancia activa en el sistema del conector a la red social (de tipo `ConectorRedSocial`) y proporcionar un método de acceso a la misma a todo el cliente.
  Esta clase factoría se encarga de la creación de dicha instancia de conexión con la red social de forma que, para cambiar el tipo de servicio, bastaría con modificarla para que instancie el conector del nuevo servicio.
  Además, respetando el patrón de diseño *Factory*, la visibilidad de la implementación concreta del conector a la red social (`ConectorFacebook.java` en nuestro caso) se restringe únicamente al paquete donde se encuentre la clase factoría.
  Con esto conseguimos que, para el resto del cliente, las clases concretas de conexión a las redes sociales o servicios ni siquiera "existan".

[7] http://en.wikipedia.org/wiki/Factory_method_pattern

4.3.4. Extracción de datos de la red social

En la interfaz `ConectorRedSocial.java` existe un método denominado `sincronizarDatosRedSocial()` que se encarga de la recuperación de datos de la red social o servicio empleado y su sincronización con el servidor.
Todo este proceso se llevará a cabo siempre en segundo plano y de forma transparente al usuario.

Se tratarán de recuperar todos los datos posibles que ofrezca la red social concreta. Aquellos que no estén disponibles pueden no ser sincronizados y el sistema Madrid Live seguirá funcionando con normalidad. Esta flexibilidad es de gran importancia para permitir la migración a otros servicios (p.ej: Twitter) o incluso la integración simultánea de varios (con el fin de recopilar más información sobre el usuario).

Para la recolección de la información, hemos dividido el método `sincronizarDatosRedSocial()` en subfunciones que se encargan, cada una de ellas, de recuperar un tipo de dato concreto. Por ejemplo: tendremos un método para recoger la lista de amigos, otro para los mensajes privados, etc. En `sincronizarDatosRedSocial()` lo que se hace es lanzar todos estos métodos más simples en paralelo de forma que puedan solicitar la información de forma independiente y la sincronicen con el servidor una vez hayan terminado.

Atendiendo a nuestra implementación concreta, es importante mencionar que, para obtener los datos de Facebook, hacemos uso de consultas FQL[8] dirigidas al *endpoint* apropiado con las credenciales de acceso del usuario que se han almacenado tras el inicio de sesión. Además, para facilitar el tratamiento y formateo de las respuestas recibidas, se ha creado un gestor en `ParserRespuestasFacebook.java` que se encarga de recibir las respuestas del *endpoint* de FQL y formatearlas adecuadamente para ser enviadas en las conexiones con la capa de servicios del servidor de Madrid Live.

4.3.5. Sistema de geolocalización

El sistema de geolocalización de Madrid Live se basa en la continua monitorización de la posición del usuario y su sincronización con el servidor.

Antes de decantarnos por este tipo de sistema estudiamos otras opciones como:

- **API de Facebook**: aunque la red social almacena siempre las últimas posiciones conocidas de los usuarios y sus amigos, es bastante restric-

[8] https://developers.facebook.com/docs/reference/fql/

tiva a la hora de facilitar esta información. Por ejemplo, en ocasiones, si se solicita la posición del usuario con la sesión activa devuelve simplemente que está en España o en la Comunidad de Madrid.
Dado el escaso grado de precisión obtenido, esta opción fue descartada por completo.

- **Google Latitude**: parecía una buena opción ya que proporcionaba toda la información que necesitábamos. Por desgracia, este servicio fue anulado hace algún tiempo, lo que nos obligó a descartarlo también.

- **GPS y notificaciones PUSH**: Android cuenta con un sistema propio de notificaciones PUSH denominado Google Cloud Service o simplemente GCM[9] que permite enviar datos desde el servidor a los dispositivos móviles.
  Esta herramienta, combinada con la información del GPS sobre la posición del usuario, sería de gran utilidad ya que permitiría notificar a los dispositivos móviles cada vez que se detectasen amigos cercanos.
  El principal problema fue que el tiempo dedicado a la integración de GCM no compensaría las ventajas frente al modelo actual (principalmente, reducción de conexiones a servidor). Además, en ocasiones este servicio puede no funcionar correctamente o tener más latencia de la esperada, lo cual podría generar problemas en un sistema de geolocalización que debe ser lo más preciso posible.

- **GPS y sincronización con servidor**: finalmente optamos por este modelo ya que ofrecía todo lo necesario para Madrid Live y además resultaba bastante simple de comprender y sencillo de implementar.
  A grandes rasgos, lo que se hace es monitorizar la posición del usuario y sincronizarla con el servidor. Cuando el sistema necesita conocer los amigos cercanos, simplemente envía otra conexión al servidor y éste le indica las últimas posiciones conocidas de los mismos.

La monitorización de la posición del usuario corre a cargo del servicio `ServicioPosicionamiento.java`, que se encuentra suscrito a notificaciones en cambios de la posición del usuario que llegan de los servicios ofrecidos por Android. En concreto, se trata de tomar la posición del GPS y de la red a la que se encuentra conectado el usuario, recogiendo siempre la más precisa de ambas.

Para evitar malgastar batería del terminal (un recurso muy importante en dispositivos móviles) la monitorización de la posición no es continua, sino que se realiza una actualización cuando el usuario se ha desplazado, al menos, 100 metros o bien cada 15 minutos si no se ha detectado ningún tipo de

[9] http://developer.android.com/google/gcm/index.html

desplazamiento (lo que permite al servicio de GPS desconectarse automáticamente al no estar en uso).

Cuando se ha recuperado una nueva posición conocida del usuario, ésta se almacena de forma local (para accesos desde el cliente) y remota (en la base de datos de servidor). El almacenamiento local permite, por ejemplo, mostrar la posición actual del usuario en el mapa de la sección de recomendación sin tener que consultar de nuevo con el servicio de geolocalización o el servidor.

Hasta aquí tenemos toda la parte que se encarga de gestionar la monitorización de la posición del usuario y sincronización con el servidor. Ahora, si el cliente necesita conocer sus amigos cercanos (para que se agreguen automáticamente al plan de ocio) solamente tendrá que conectarse al servidor y solicitar las últimas posiciones conocidas de sus amigos. Con este dato y la posición del usuario, se detectan los amigos que están en un radio determinado (220m) de cercanía. Este radio es amplio para tener en cuenta tanto posibles imprecisiones del GPS o el monitor de redes (unos 10 metros de margen) como el margen de actualización de 100 metros mencionado antes (por ahorro de batería).

4.3.6. Captura de imágenes

A continuación se hablará de un módulo creado para Madrid Live que se integra con otro proyecto independiente de Madrid Live. Se trata de la detección automática de la aceptación o rechazo de un plan de ocio. La integración de este módulo no está completa y se encuentra en fase experimental, pero la comentaremos muy brevemente de todos modos.

En primer lugar, decir que se cuenta con un proyecto externo que permite reconocer estados de ánimo a partir de las fotos de los rostros de los usuarios, cuya aplicación a Madrid Live sería que, tras mostrar un plan al usuario y dejar que lo visualizase, se capturase una foto de la cara del mismo y se obtuviese, con ayuda de este proyecto externo, el estado de ánimo o expresión del rostro del usuario. Con ello podríamos detectar si al usuario le ha gustado o no el plan recomendado y, además, el grado de satisfacción en función de la foto capturada.

Ahora mismo, como se ha mencionado, todo esto se encuentra en fase experimental y, aunque el reconocimiento de expresiones a partir de la foto sí da buenos resultados, no se ha podido verificar la tasa de aciertos al acoplarlo a Madrid Live y usarlo para aceptar o rechazar planes automáticamente.

Lo que sí hace ahora mismo Madrid Live es almacenar una pequeña base de imágenes en la que se guardan las capturas de pantalla realizadas etiquetadas según se haya aceptado o rechazado el plan.

Para la gestión de dichas capturas se hace uso de la clase `GestorCamara.java` que se encarga de la captura y guardado de las fotos tomadas a través de los siguientes pasos:

1. En caso de que el terminal disponga de cámara frontal, al entrar en la pantalla de visualización del plan de ocio recomendado se inicializa y abre dicha cámara.

2. Cuando el usuario acepta o rechaza el plan, se captura una imagen con la cámara frontal.

3. Esta foto se almacena en la base de imágenes local dentro del directorio apropiado ("Aceptados" o "Rechazados").

4. Se liberan los recursos asociados a la cámara frontal.

A todo esto, por supuesto, habría que sumarle la integración con este proyecto externo para la detección del estado de ánimo a partir del rostro del usuario. No obstante, esta parte queda abierta para una posible línea de investigación futura.

Conclusiones

En este capítulo hemos detallado la arquitectura y funcionamiento del sistema Madrid Live como complemento a la información ofrecida en el Capítulo 3.

Hemos comenzado explicando la arquitectura general de Madrid Live para continuar hablando sobre las funcionalidades básicas del sistema (recomendaciones, gestión de datos, etc.) correspondientes al núcleo del mismo. De ahí hemos pasado a ofrecer una breve visión de los detalles de estructura y diseño de la capa de servicios (API), información que puede completarse consultando el Apéndice B. También hemos comentado todos los detalles más relevantes relativos al diseño de la aplicación Android como puede ser el funcionamiento del servicio de geolocalización.

A continuación veremos los resultados de un experimento de validación llevado a cabo para el sistema Madrid Live, así como las conclusiones generales del proyecto y las líneas de investigación futuras.

Capítulo 5

Validación experimental

En este capítulo mostraremos los resultados de un experimento de validación llevado a cabo para el sistema así como las conclusiones obtenidas del mismo.

5.1. Diseño del experimento

Para poner a prueba todas las funcionalidades de Madrid Live se ha llevado a cabo un experimento de validación del sistema desplegando una versión beta del mismo en un servidor público y repartiendo el cliente Android a un grupo de usuarios. También se ha adjuntado a este cliente un pequeño tutorial muy básico de las diferentes funcionalidades de Madrid Live y una sencilla encuesta acerca de las mismas. Las preguntas de dicha encuesta estaban preparadas para que el usuario evaluase las diferentes funcionalidades de Madrid Live puntuándolas con un valor entre 1 y 5, donde 1 significaría que la funcionalidad no ha resultado útil o no ha funcionado correctamente y 5 que ha agradado mucho al usuario.

El manual incluía, entre otras anotaciones, una serie de puntos básicos que se requerían al usuario antes de rellenar la encuesta final:

- *Instalación de la aplicación Android e inicio de sesión con Facebook.*
- *Realización de los tests de preferencias.*
- *Búsqueda de, al menos, un plan individual.*
- *Búsqueda de, al menos, un plan grupal.*
- *Valoración de actividades.*

Al margen de estos puntos, se ha dejado un periodo de 3 semanas aproximadamente para que los usuarios pudiesen probar la aplicación de Madrid Live libremente.

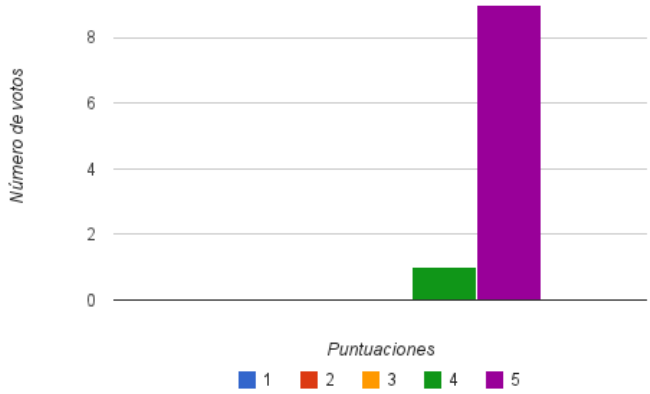


Figura 5.1: Puntuaciones de la encuesta sobre el uso de plataforma móvil

5.2. Resultados del experimento

Una vez finalizadas las pruebas, se recogieron tanto los datos de las encuestas, como el *feedback* recibido personalmente como la información de servidor (estudio de la base de datos para extracción de información relevante). A continuación vamos a comentar los resultados obtenidos para las diferentes funcionalidades de Madrid Live.

Por un lado, el hecho de que el cliente estuviese implementado en Android, como podemos apreciar en los resultados de la encuesta mostrados en la Figura 5.1, fue algo que gustó especialmente a los usuarios. El principal punto a favor ha sido que se trata de una tecnología muy actual y que tiene un *engagement* considerable sobre todo en el público más joven.

Hoy en día los dispositivos móviles son usados con muchas y muy variadas finalidades: chats con amigos, navegación por Internet, envío de correo personal, etc. Por ello, cada vez es más frecuente que existan aplicaciones que ofrezcan servicios concretos de forma mucho más orientada al dispositivo móvil. Por ejemplo: tenemos una aplicación Android para el sistema de correo Gmail aunque éste también es accesible a través del navegador web del dispositivo. Esto ha jugado una baza muy importante porque, si tengo un dispositivo móvil que puede recoger información de contexto (posición, amigos cercanos, etc.), ¿por qué necesitaría entrar a través de un navegador web para rellenar tediosos formularios?

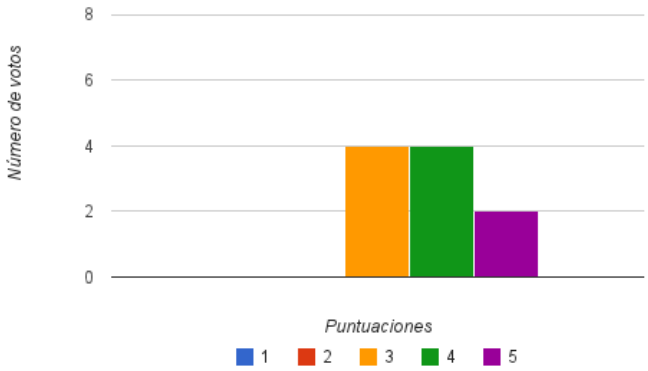


Figura 5.2: Puntuaciones de la encuesta sobre la utilidad general

En lo que respecta a la parte más visual y relacionada con la experiencia de usuario, la opinión general es que la aplicación resulta muy sencilla de utilizar y bastante intuitiva. El principal cambio solicitado por los usuarios está directamente relacionado con los tests de preferencias (lo que confirma las líneas de trabajo futuro que se verán en el Capítulo 7).
Aunque la utilidad de los tests está clara, sí es necesario darles una vuelta de nuevo y rediseñarlos para que sean más amigables para el usuario. Por ejemplo: la lista de géneros de películas es grande y buscar otro modelo de test más simple supondría un beneficio considerable.
En la Figura 5.2 podemos ver las votaciones de la encuesta sobre si el usuario usaría o no Madrid Live para solicitar planes de ocio y en la Figura 5.3 los votos dados a la parte relacionada con el diseño y estructura de las funcionalidades (experiencia de usuario).

Otro de los aspectos que han tenido una buena acogida ha sido la integración con una red social como Facebook (ver Figura 5.4). La principal ventaja vista por los usuarios ha sido la posibilidad de que se hayan podido usar todos los datos de la misma (p.ej.: la lista de amigos) sin necesidad de tener que crearlos de nuevo en Madrid Live. Por supuesto, obtener la lista de amigos no es la principal ventaja de la integración con una red social, sino la posibilidad de obtener datos mucho más precisos sobre el círculo social completo del usuario, lo cual ayudará en gran medida en las recomendaciones grupales.

También ha tenido una muy buena aceptación el sistema de geolocalización (ver Figura 5.5). De nuevo, a los usuarios les ha gustado la funcionalidad

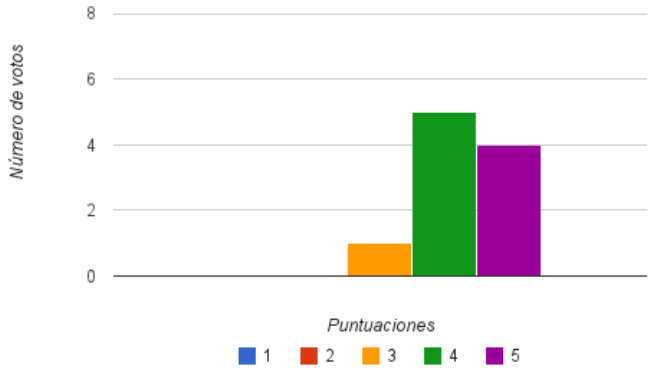


Figura 5.3: Puntuaciones de la encuesta sobre la experiencia de usuario

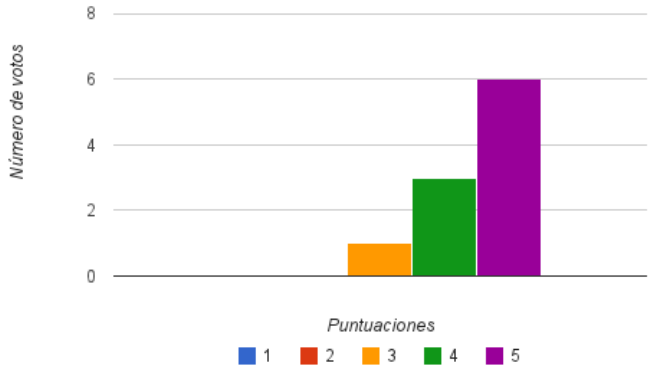


Figura 5.4: Puntuaciones de la encuesta sobre la integración con Facebook

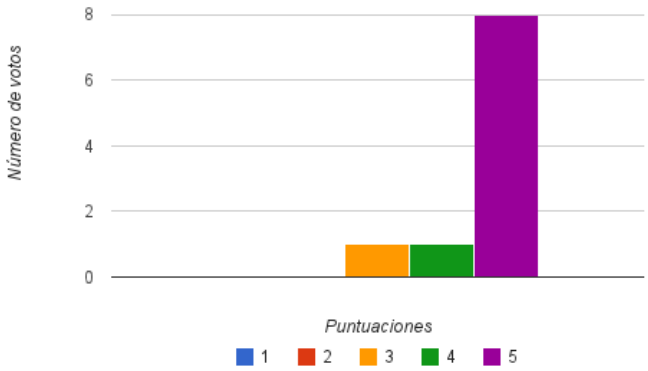


Figura 5.5: Puntuaciones de la encuesta sobre el sistema de geolocalización

de Madrid Live que se encarga de detectar los amigos cercanos y agregarlos al plan.
De hecho, y enlazando un poco con todos los puntos anteriores, uno de los aspectos más atractivos del cliente Android ha sido la capacidad general de recopilar información del entorno (contexto) evitando así que el usuario tenga que introducir datos manualmente. Por ejemplo, como ya hemos visto en la Sección 3.5, cuando se solicita una recomendación el sistema recoge toda la información posible del contexto actual del usuario y la fija automáticamente en la pantalla de recomendación, lo que supone una gran ventaja ya que el usuario puede solicitar un plan de ocio para el mismo momento de la recomendación sin tener que introducir prácticamente ninguna información. Los resultados de estas pruebas son de gran importancia ya que confirman que la línea seguida por el proyecto de Madrid Live ha sido la adecuada. Recordemos que uno de los ejes centrales siempre ha sido la recopilación automática de información de contexto y su inclusión en el sistema de recomendación.

Por último, en lo que a las recomendaciones en sí se refiere, el sistema ha conseguido una elevada tasa de aciertos. Las Tablas 5.1 y 5.2 muestran un resumen aproximado del número de recomendaciones individuales y grupales recogidas junto con los planes que han sido aceptados y rechazados y sus valoraciones medias.

Como se puede ver tanto en las tablas anteriores como en las Figuras 5.7 y 5.8, la tasa aciertos para los planes individuales ha sido bastante elevada

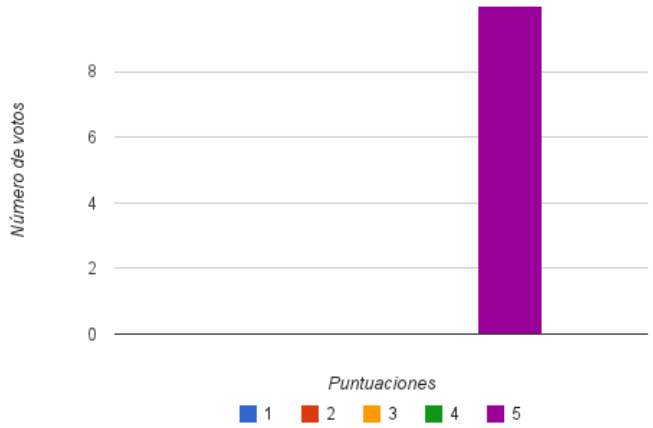


Figura 5.6: Puntuaciones de la encuesta sobre el sistema de gestión de grupos

Número de recomendaciones individuales solicitadas	72
Número de recomendaciones individuales aceptadas	60
Número de recomendaciones individuales rechazadas	12
Tasa de aciertos en recomendaciones individuales	83,3 %
Tasa de fallos en recomendaciones individuales	16,7 %
Valoración media de las actividades y plantillas individuales	4,35 / 5

Tabla 5.1: Datos del experimento de recomendaciones individuales

(83,3 %). Además, la puntuación media de las actividades recomendadas en dichos planes también es bastante resaltable, lo que indica que tanto el perfil de preferencias del usuario generado a partir de los tests de preferencias como la información de contexto agregada a la recomendación llevan una línea de desarrollo correcta y constituyen potentes fuentes de información para el sistema de recomendación.

Por otro lado, los resultados para las recomendaciones grupales han sido similares: una alta tasa de aciertos (81,63 %) y puntuaciones medias para las actividades más que aceptables. La diferencia frente a las recomendaciones individuales es que, en este caso, tratamos con grupos de usuarios, lo cual supone diferentes gustos, personalidades, etc. Los resultados obtenidos para esta parte del experimento demuestran que el sistema ha mantenido una calidad de recomendación similar al caso de usuarios individuales, lo que confirma que la integración con una red social y la creación de un perfil social muy preciso de los usuarios proporciona al sistema una muy necesaria

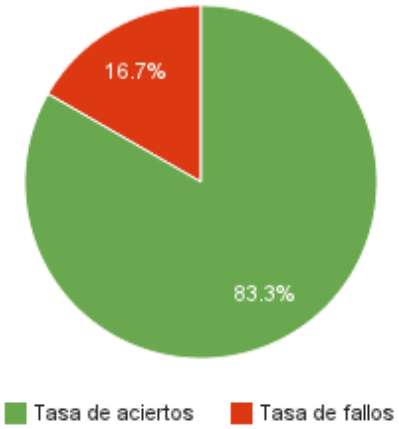


Figura 5.7: Tasa de aciertos y fallos en recomendaciones individuales

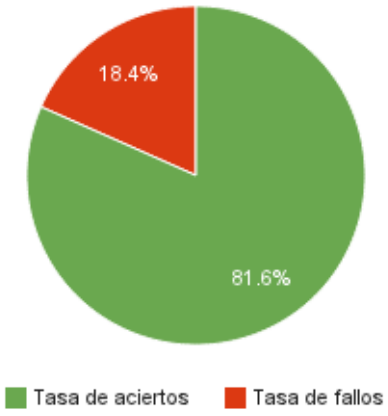


Figura 5.8: Tasa de aciertos y fallos en recomendaciones grupales

Número de recomendaciones grupales solicitadas	49
Número de recomendaciones grupales aceptadas	40
Número de recomendaciones grupales rechazadas	9
Tasa de aciertos en recomendaciones grupales	81,63 %
Tasa de fallos en recomendaciones grupales	18,37 %
Valoración media de las actividades y plantillas grupales	4,1 / 5

Tabla 5.2: Datos del experimento de recomendaciones grupales

fuente de información para mejorar las recomendaciones realizadas a grupos.

En este caso, además, hemos podido comprobar que los valores de *trust* (ver Sección 3.4.7) obtenidos durante las pruebas valoraban correctamente las relaciones de confianza entre cada par de usuarios. Aquellos con gran número de interacciones en la red social (Facebook) han mostrado un valor de *trust* mucho mayor que los usuarios que se habían agregado entre sí solamente para alguna prueba (es decir, sin contacto en la red social).

Como añadido a esto último, aunque se han realizado varias pruebas con diferentes tipos de grupos, las más resaltables son aquellas en las que existía un núcleo del grupo consolidado (amigos en Facebook con bastante interacción) al que se agregaban algunos usuarios completamente nuevos y ajenos a dicho núcleo. En estos casos, en general, la recomendación se veía claramente enfocada a los gustos del núcleo principal del grupo (valoraciones más altas de actividades) frente a los usuarios ajenos (puntuaciones ligeramente menores).
Este tipo de pruebas fueron realizadas especialmente para comprobar la utilidad y eficacia de la integración del valor de confianza o *trust* en las funciones de agregación grupales. En la implementación actual de Madrid Live la principal función de este valor de *trust* era, justamente, variar la función de agregación media (la más justa para todos los usuarios) haciendo que se favoreciese más al núcleo del grupo.

Para terminar, en lo que a la encuesta realizada se refiere, los resultados obtenidos en el apartado de calidad de recomendaciones individuales y grupales concuerdan con la información extraída de las pruebas. Ha habido una valoración media de 4,2 puntos sobre 5 (Figura 5.9) para las recomendaciones individuales y 4 puntos para el caso grupal (Figura 5.10).

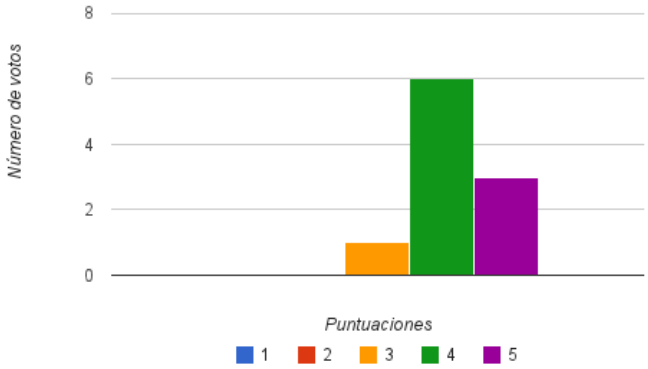


Figura 5.9: Puntuaciones de la encuesta sobre las recomendaciones individuales

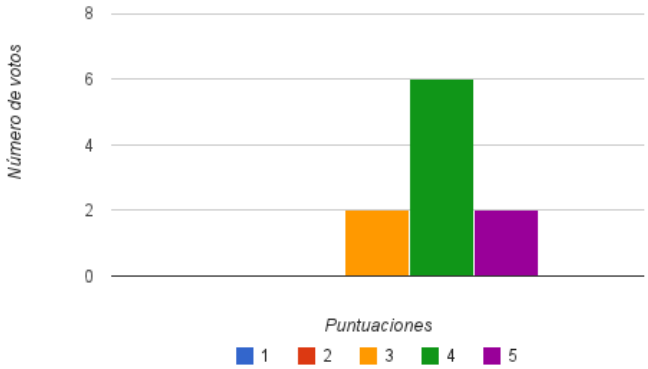


Figura 5.10: Puntuaciones de la encuesta sobre las recomendaciones grupales

5.3. Conclusiones del experimento

Los resultados obtenidos en el experimento de validación han sido realmente buenos. Las nuevas funcionalidades (geolocalización, cliente Android, etc.) han tenido una buena acogida y, además, la calidad de las recomendaciones ofrecidas ha sido mejor de la esperada. Se ha visto una clara mejoría en las recomendaciones con respecto al proyecto original que se basaba en información más básica y estática.
Por supuesto, aún queda trabajo por hacer para seguir perfeccionando el sistema, pero los resultados de las pruebas muestran que las líneas de desarrollo e investigación seguidas durante este año para el proyecto de Madrid Live han sido las adecuadas y han dado muy buenos resultados.

Capítulo 6

Conclusiones del proyecto

En el Capítulo 5 hemos descrito el experimento de validación realizado junto con las conclusiones obtenidas del mismo y ahora vamos a comentar ahora las conclusiones generales de todo el proyecto.

En primer lugar, en lo que al cumplimiento de objetivos en el proyecto se refiere, se han cerrado todos los puntos previstos:

- **Plataforma móvil**. El primer gran cambio llevado a cabo sobre el proyecto original ha sido la migración del cliente web a dispositivos móviles (concretamente Android), lo que nos abrió las puertas por completo a la ampliación del sistema de recomendación agregando información contextual acerca del momento de la recomendación (posición, amigos cercanos, etc.).
- **Sistema de geolocalización**. A pesar de que actualmente los dispositivos móviles ya integran tecnologías como el GPS o la detección de la posición por las redes a las que se conecta el terminal, ha sido necesario crear un nuevo módulo que nos permitiese no solo detectar la posición del usuario sino localizar a sus amigos cercanos.
  Esta funcionalidad también era uno de los objetivos principales para el sistema ya que era necesario contar con algún sistema que permitiese conocer tanto la posición del usuario como la de los amigos cercanos.
- **Integración con red social**. Otra de las incorporaciones fue la conexión a Madrid Live a través de una red social (Facebook en nuestro caso), cuya mayor ventaja ha sido, sin duda, poder extraer toda la información posible de dicha red social para generar nueva información para el perfil de usuario (contexto social) y mejorar así, sobre todo, las recomendaciones grupales.
  Durante el experimento de validación realizado hemos podido probar cómo la inclusión de este factor social afectaba claramente a las recomendaciones para grupos.

- **Información de contexto**. Otro de los principales objetivos y puntos centrales del proyecto ha sido la incorporación de toda la información de contexto posible para automatizar y mejorar las recomendaciones, lo que ha resultado de gran importancia de cara a la línea de investigación ya que se ha podido ver la mejoría en las recomendaciones respecto a la información más estática manejada en el proyecto original.
  Para su cumplimiento se ha hecho uso tanto de las funcionalidades de Android para recopilar toda la información posible del entorno como de los sistemas de geolocalización y la integración con una red social mencionados antes.

- **Mejoras al perfil de usuario**. Relacionadas con este objetivo se han hecho varias aportaciones. Por un lado, se ha mejorado la gestión del perfil de usuario del proyecto base (gustos del usuario, valoraciones básicas, etc.) afinando la actualización de los datos y agregando algunos adicionales al sistema de recomendación. Por ejemplo, ahora todas las actividades pueden ser valoradas por el usuario y dichas valoraciones se tienen en cuenta a la hora de realizar recomendaciones.
  Por otro lado, se han incorporado nuevos atributos al perfil de preferencias del usuario. Al margen de todo aquello relacionado con el contexto a la hora de la recomendación o el contexto social, se han incorporado nuevas características como la preferencia de los usuarios a desplazarse, a realizar un plan fuera del horario previsto o sus gustos por cada tipo de actividad recomendada.

- **Mejoras en las recomendaciones grupales**. Para lograrlo, al margen de todos los beneficios obtenidos de la información de contexto, una de las mejoras pasaba por la incorporación de funciones de agregación más precisas.
  Como se ha mencionado en la Sección 3.4.7, se ha agregado a Madrid Live una nueva función de agregación al margen de las básicas que tiene en cuenta el valor de confianza o *trust* de los miembros de un grupo entre ellos.

- **Nueva actividad de ocio**. Por último, para ampliar un poco el catálogo de recomendación, otro de los objetivos principales era la inclusión de nuevas actividades de ocio, lo que se ha conseguido tras introducir cines y películas. Esta nueva actividad, junto con las demás del proyecto base, han conseguido dar bastante juego a la hora de realizar recomendaciones.

Dejando al margen los cambios realizados, debemos hablar también de todos los conocimientos adquiridos durante la realización del proyecto.

Los sistemas de recomendación son un tema muy de moda. Pueden verse módulos recomendadores en la mayoría de servicios web (Amazon, Kiabi, etc.) e incluso sistemas cuya base es la recomendación de productos (Tripadvisor, Fever, etc.). El uso actual tan elevado de este tipo de sistemas ha despertado mucho nuestro interés de cara al proyecto y, sin duda, a lo largo de la realización del mismo hemos adquirido bastantes conocimientos relacionados con sistemas de recomendación.

Otra de las técnicas aprendidas ha sido la extracción de datos del círculo social de un usuario para crear un perfil social del mismo. Por ejemplo, lo que ya comentamos en varias ocasiones de obtener el nivel de confianza de un usuario con sus amigos, lo cual nos permite, entre otras cosas, detectar líderes de opinión en los grupos.

No podemos olvidar tampoco el uso de la tecnología Android para la implementación del cliente de Madrid Live y la recolección de información de contexto. Hablamos de una tecnología de vanguardia con un gran uso en la actualidad.

Todo esto es importante ya que muestra que el proyecto no ha sido útil solamente en lo relacionado con el ámbito académico y de investigación sino que nos ha proporcionado conocimientos muy valiosos de cara al mundo profesional.

Por último, y como se detallará en la siguiente sección, Madrid Live no es solo un proyecto que haya sido aprovechado este año sino que permite una gran cantidad de líneas de investigación y desarrollo futuras. Teniendo en cuenta que el proyecto abarca tecnologías de vanguardia y temas de investigación actual, creemos que la continuidad del mismo sería un refuerzo muy positivo tanto a las labores de investigación como al aprendizaje de las tecnologías empleadas. Por ejemplo, se puede aprovechar todo el funcionamiento actual para incorporar nuevos elementos al perfil de usuario (marcadores emocionales o psicológicos, rangos de edad, etc.) para ver cómo reacciona el sistema de recomendación al hacer uso de los mismos.

En resumen, para nosotros la continuidad del proyecto de Madrid Live iniciado en el curso 2012-2013 ha sido realmente positiva, sobre todo, desde el punto de vista del aprendizaje. Estamos muy contentos de las mejoras introducidas en el sistema así como de los resultados de las pruebas locales y el experimento de validación. Esperamos también que Madrid Live no sea un proyecto que muera este año y animamos a todos los interesados a continuar con él.

Capítulo 7

Líneas de trabajo futuro

Aún tras todo el camino recorrido en el presente curso con Madrid Live, hay muchas líneas abiertas para la continuación del proyecto actual. A continuación daremos una visión global de varias posibles líneas de trabajo futuro, centrándonos primero en aquellos cambios orientados a mejorar las funcionalidades ya existentes para terminar hablando de nuevas líneas de desarrollo así como de la incorporación de nuevos módulos.

En primer lugar, una de las posibles mejoras pasaría por la *ampliación o cambio de las fuentes de información* de las actividades de ocio.
Como en todo sistema de recomendación, la calidad de los datos de los elementos a recomendar es de vital importancia, por lo que, manejar información lo más completa posible es una gran ventaja ya que permite ofrecer al usuario detalles más útiles acerca de los productos recomendados y, al mismo tiempo, permite jugar con una cantidad mayor en el módulo de recomendación, con lo que se puede afinar mucho más la búsqueda en función de las preferencias del usuario.
Por otro lado, al margen de la cantidad de información sobre los productos a recomendar, también tenemos la calidad de la misma. De nuevo, este es otro factor de gran importancia para que tanto la recomendación como la presentación de los resultados sean de la mejor calidad posible.
En general, existe una estrecha relación entre los datos que maneja el sistema de recomendación y los resultados ofrecidos por el mismo, lo que nos llevaría a una posible mejora de las fuentes de información actuales.

Ahora mismo Madrid Live cuenta con aquellas fuentes de información que nos han permitido recopilar datos con una calidad aceptable sin requerir gran tiempo de investigación o implementación. Estas fuentes, que se han detallado en la Sección 3.2.2, pueden ser fácilmente ampliables o reemplazables. Por ejemplo: la información sobre paseos que existe actualmente es bastante limitada. Poder contar con un módulo nuevo que genere paseos a

partir de una colección de puntos de interés dados (trabajo con grafos) sería de gran ayuda para ampliar la variedad del catálogo de recomendación actual.
Por otro lado, la información sobre ciertas actividades, en ocasiones, no está completa: no todas tienen fotos asociadas, algunas tienen descripciones imprecisas, etc.

Las dos posibles mejoras relacionadas con toda esta parte pasarían o bien por el reemplazamiento de las fuentes de información actuales o bien por el acoplamiento de otras nuevas.
En el primer caso bastaría con eliminar uno o varios de los servicios de información actuales y reemplazarlos por otros nuevos más completos. Por ejemplo, si la información sobre películas obtenida a través de la guiadelocio.com resultase ser de baja calidad, podría buscarse algún otro servicio de información de películas como sustituto. Este cambio estaría principalmente orientado a la calidad de la información.
Otra de las opciones es mantener los servicios actuales pero complementar la información con ayuda de algún otro. Por ejemplo, si hay pocos parques en la base de datos actual, se podrían ampliar buscando en más de un servicio y combinando la información recopilada. En este caso, el cambio iría más orientado a la cantidad de información.

Como ayuda para todo esto, en su momento se investigó una web del ayuntamiento de Madrid (http://datos.madrid.es/portal/site/egob/) que ofrece información sobre diversas actividades (hoteles, aeropuertos, conciertos, etc.). El principal problema de esta web radica en que es algo reciente y la estructura de la información o bien difiere bastante de la actual de Madrid Live o bien no es muy accesible. No obstante, sí convendría seguir revisando este servicio ya que, en lo que a información se refiere, contiene gran cantidad de actividades que se actualizan periódicamente (lo cual es muy positivo).

Otra de las líneas de desarrollo, más relacionada con *el cliente Android*, pasaría por mejorar la usabilidad y experiencia de usuario. Ahora mismo el cliente es bastante básico y contiene las funcionalidades necesarias para poder hacer uso del sistema, por lo que una línea de trabajo basada en la aplicación Android pasaría por realizar un estudio detallado de la interfaz, experiencia de usuario y usabilidad e implementar un nuevo diseño para el cliente. Con ello se lograría crear un Madrid Live mucho más atractivo y permitiría, incluso, llevarlo a versiones de producción. Por ejemplo, ahora mismo los tests de preferencias pueden resultar bastante tediosos. Dividirlos en varias secciones y hacerlos mucho más amigables sería de gran ayuda para animar al usuario a completarlos y perfeccionar así el perfil de preferencias.
También se podría realizar una migración de la aplicación nativa actual de

Android a una aplicación web[1] que permitiría una integración mucho más simple con otras plataformas (iOS, web, etc.).

En lo que al *sistema de recomendación* se refiere, hay múltiples mejoras o líneas de desarrollo que se pueden seguir. A continuación citamos algunas de ellas:

- ***Actividades:*** una de las posibles mejoras es la inclusión de nuevas actividades. Por ejemplo, algo de lo que carece el sistema actual son actividades de ocio nocturnas. Incluírlas supondría una gran ventaja de cara al uso del público más joven.

- ***Planes para varios días:*** otra de las limitaciones del sistema actual es que las recomendaciones solicitadas son siempre para el día actual. Ampliar esto para permitir realizar planificaciones de varios días sería, de nuevo, una ventaja muy buena. Este punto enlazaría con el anterior ya que, posiblemente, la inclusión de hoteles en la recomendación fuese necesaria.

- ***Zona de recomendación:*** actualmente Madrid Live solo realiza recomendaciones para la zona de Madrid. Incluyendo nuevas fuentes de información podríamos conseguir ampliar el sistema para cubrir nuevas ciudades (p.ej.: Barcelona, Valencia, etc.) o incluso comunidades enteras.

- ***Rechazo de recomendaciones:*** en la implementación actual, cuando un usuario rechaza un plan de ocio, no se guarda ningún tipo de información acerca del plan concreto rechazado. Por supuesto, sí se actualizan las preferencias o gustos del usuario en cuestión. Otra posible mejora pasaría por crear un módulo adicional que se encargase de analizar los planes rechazados por el usuario de forma que se tuviese esto en cuenta durante el ciclo CBR para evitar así repeticiones no deseadas.

- ***Perfiles de grupos:*** como Madrid Live está diseñado tanto para planes individuales como grupales, y ya existe un perfil bastante detallado de los gustos de los usuarios individuales, sería interesante contar con otros tipos de perfiles orientados a grupos. En ellos se considerarían atributos que pertenezcan al conjunto y no a los individuos por separado. Por ejemplo: se podrían tener en cuenta la media y desviación de las edades de los miembros, perfiles de personalidad, etc. para catalogar los planes y usar esta nueva información en la recomendación a futuros grupos.

[1] http://developer.android.com/guide/webapps/index.html

Como se puede ver, en lo que al sistema de recomendación se refiere, hay un sinfín de mejoras posibles. Este es otro de los puntos a favor para dar continuidad al proyecto actual.

Para ir terminando, en la Sección 4.3.6 se habló de un módulo de *detección automática de la aceptación o rechazo de un plan*, que se encuentra en estado de pruebas y, por lo tanto, con funcionalidades incompletas. Sería muy interesante tener una línea de trabajo futuro que llevase dicho módulo a una versión completamente funcional, lo que estaría estrechamente relacionado con la experiencia de usuario ya que ahora no sería necesario que éste aceptase o rechazase el plan sino que Madrid Live se encargaría de detectar automáticamente esta acción y, además, podría evaluar el nivel de satisfacción en función del rostro del usuario. Además, los datos recogidos por este módulo ampliarían toda la información contextual ya presente en el sistema.

Por último, otro de los módulos que podría resultar muy interesante sería el *análisis textual de la información* de las actividades recomendadas.
Ahora mismo se cuenta con descripciones u opiniones escritas por usuarios que han realizado las actividades pero el problema es que dicha información está bastante desaprovechada ya que se usa principalmente para ofrecerla al usuario.
Contar con un módulo que sea capaz de analizar estos textos y extraer de ellos un valor de satisfacción para cada actividad sería muy positivo de cara a mejorar el sistema de recomendación. Se contaría con nueva información que se agregaría a las valoraciones manuales y las puntuaciones obtenidas de los servicios de información actuales.
Se podría ir incluso más allá recuperando información de las redes sociales acerca de las actividades a recomendar y agregándola a lo mencionado anteriormente. Por ejemplo: podríamos monitorizar constantemente la red de Twitter para obtener tweets en los que se mencione un determinado elemento (supongamos que es el Museo del Prado). A continuación, analizaríamos toda esta información para extraer datos o valoraciones sobre esta actividad concreta (por ejemplo, a través de *sentiment analysis*[2]) y los agregaríamos al sistema de recomendación actual.
Gracias a esto tendríamos información muy actualizada sobre las diferentes actividades a recomendar. Si, por ejemplo, un usuario solicita un museo y se le recomienda el Museo del Prado por encajar con sus gustos, pero resulta que justo la exposición de este mes de dicho museo es realmente mala, Madrid Live podría detectarlo a través de las publicaciones de otros usuarios en las redes sociales y evitar recomendar el Museo del Prado mientras se reciban opiniones muy negativas sobre él.

[2]http://en.wikipedia.org/wiki/Sentiment_analysis

Como se puede ver, las líneas de trabajo futuro son bastantes y muy variadas. Aquí se han comentado algunas de las posibles pero seguramente existan muchas más. Después de haber trabajado durante todo un año sobre el proyecto hemos podido ver que realmente da mucho juego. Nosotros hemos preferido centrarnos en aspectos más relacionados con investigación (inclusión de una agregación basada en *trust*, recolección de información del contexto con tecnologías móviles, etc.) pero existe un abanico bastante amplio de posibilidades en lo que a la continuación de este proyecto se refiere. Para finalizar, y a modo de curiosidad, si volviésemos atrás sobre todas las reuniones del proyecto, encontraríamos multitud de ideas relacionadas con Madrid Live que al final no han podido ser implementadas.

Apéndice A

Base de datos

A continuación mostraremos la estructura de la base de datos MySQL de Madrid Live mediante una serie de figuras, en cada una de las cuales se podrán ver las diferentes tablas con sus correspondientes atributos y relaciones con las demás. Cada figura mostrará una colección de tablas relacionadas con una determinada sección dentro del sistema.

A.1. Almacenamiento del perfil del usuario

La Figura A.1 recoge las tablas que intervienen en el almacenamiento de los datos referentes al perfil del usuario. Es decir, la colección de tablas que contiene los gustos o preferencias de cada usuario.

A.2. Almacenamiento de las plantillas realizadas por los usuarios

En la Figura A.2 aparecen las tablas que guardan la información sobre las plantillas de planes de ocio que ha realizado cada usuario. Por cada plantilla también aparecen las actividades de las que está compuesta y la fecha de realización de la misma.

A.3. Almacenamiento de los datos para la recomendación

En las figuras A.3 y A.4 se muestran las tablas que almacenan los datos necesarios para poder realizar las recomendaciones y que contienen toda la información posible acerca de cada una de las diferentes actividades de ocio.

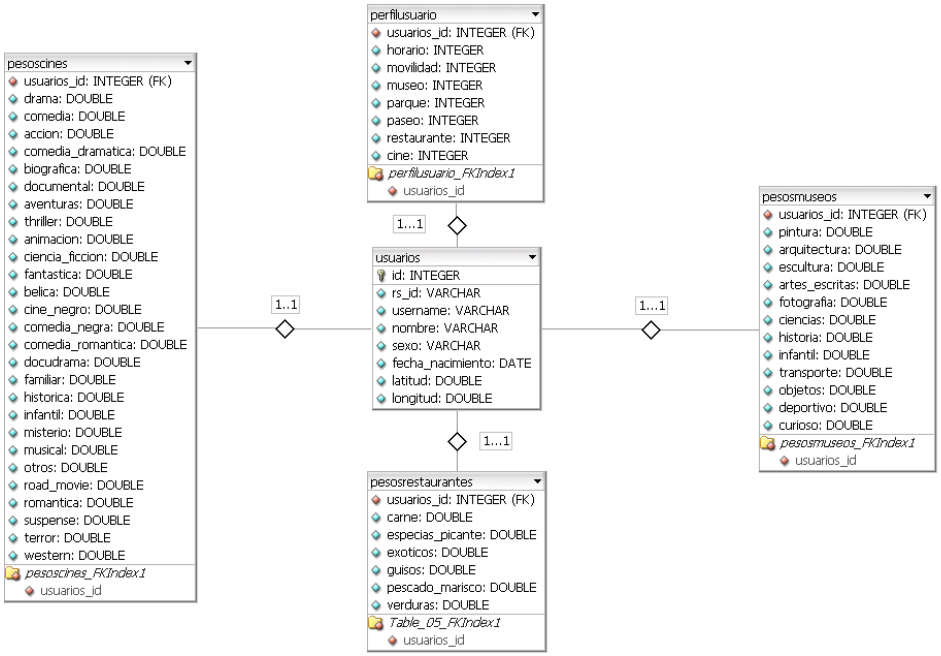


Figura A.1: Tablas para el perfil del usuario

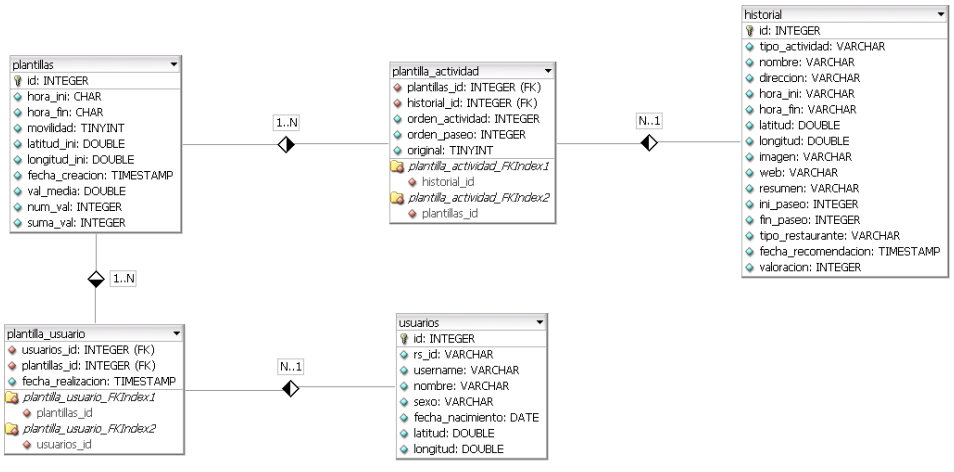


Figura A.2: Tablas para el guardado de plantillas

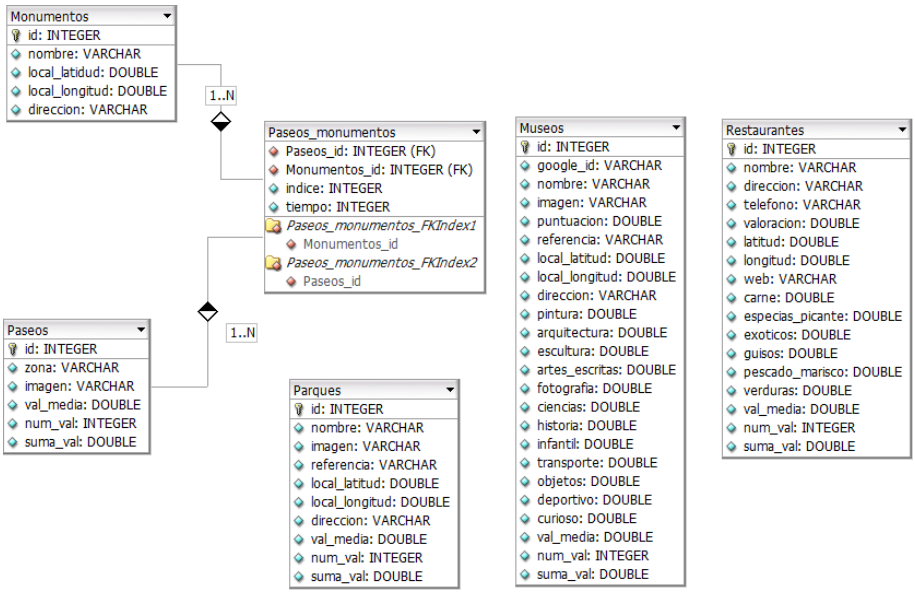


Figura A.3: Tablas utilizadas en la recomendación

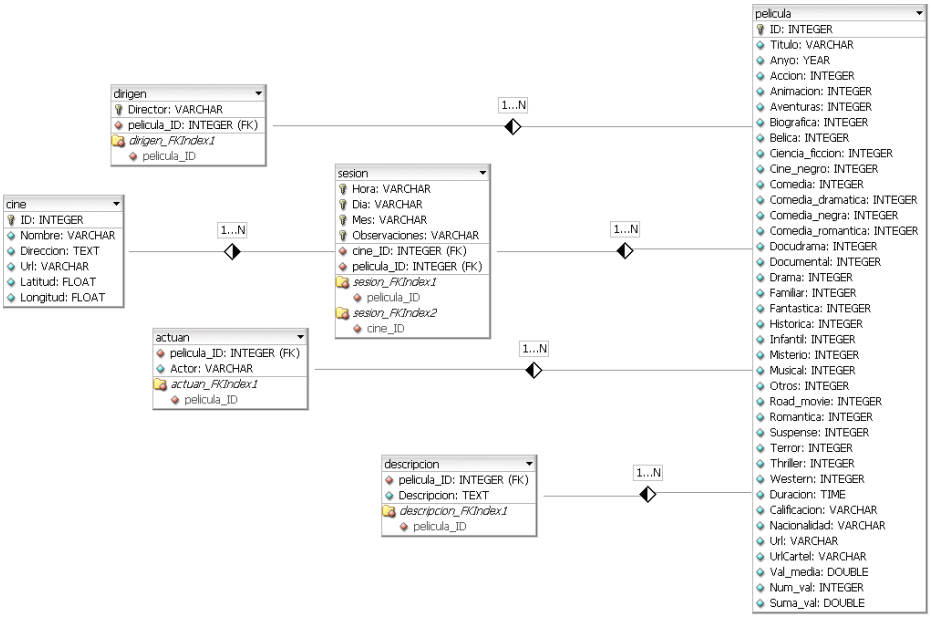


Figura A.4: Tablas utilizadas en la recomendación

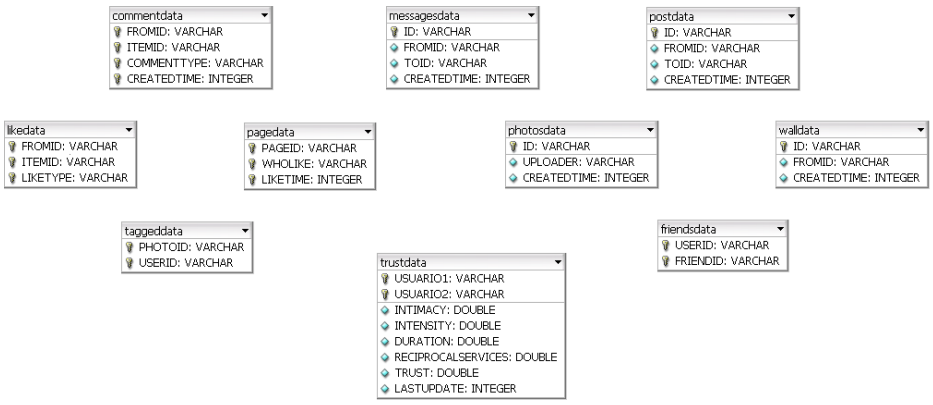


Figura A.5: Tablas utilizadas en el cálculo del trust

A.4. Almacenamiento de los datos recogidos de la red social

Las tablas que intervienen en el almacenaje de los datos recuperados de la red social han quedado recogidas en la Figura A.5. En ellas se almacena también el *trust* calculado entre dos usuarios para poder utilizarlo más tarde en la recomendación grupal.

A.5. Almacenamiento de grupos para la recomendación

La Figura A.6 representa el conjunto de tablas encargadas del almacenamiento de los grupos de amigos creados por el usuario para sucesivas recomendaciones. Este grupo de tablas recoge informacion que permite agilizar las recomendaciones cuando un usuario suele ir con el mismo grupo de amigos.

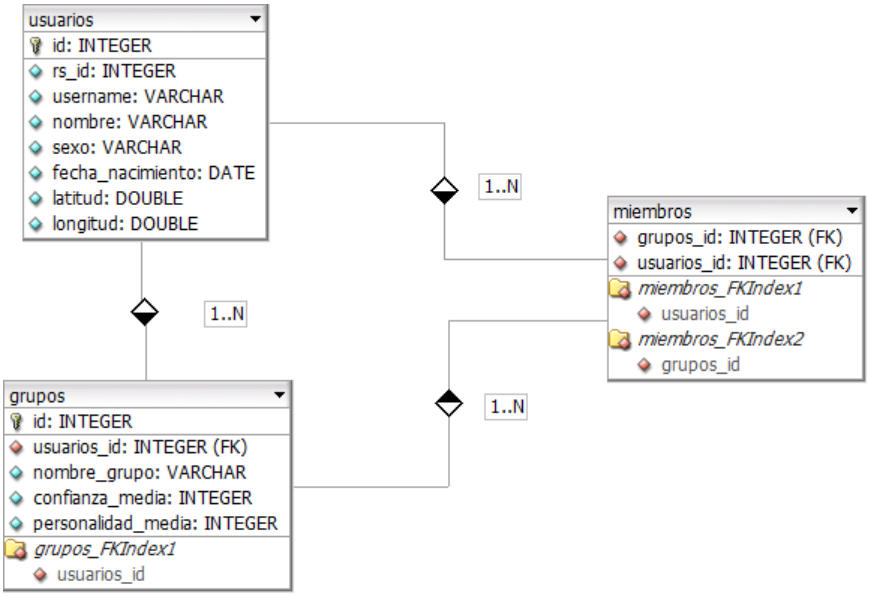


Figura A.6: Tablas utilizadas en el almacenamiento de grupos

Apéndice B

API

A continuación vamos a mostrar de forma muy esquemática todas las llamadas que ofrece la capa de servicios junto con el *Servlet* asociado.

B.1. Gestión de datos de usuario

B.1.1. Login

Crea una sesión en el servidor para el usuario si las credenciales son correctas.

- **Servlet:**
  `LoginServlet.java`
- **URL:**
  <servidor:puerto>/TurismoAPI/login.do
- **Método:**
  `GET`
- **Parámetros:**
  - *rsid:* ID único usado como identificador en la red social a la que se conecta Madrid Live.
  - *username:* identificador de usuario usado para el inicio de sesión en la red social (p.ej.: email).
- **Resultado:**
  - *mensaje:* mensaje informativo ofrecido como respuesta. Especialmente útil en caso de error.

- *status:* estado de la solicitud. "OK" si la petición se ha completado correctamente o "ERROR" si ha ocurrido algún error.
- *nuevo:* booleano que indica si el usuario que inicia sesión se ha registrado por primera vez en el sistema tras este login.

- **Ejemplo:**
  - *Conexión a:* <servidor:puerto>/TurismoAPI/login.do?rsid=1234&username=testuser
  - *Resultado:*

```
{
‘‘nuevo’’ : ‘‘true’’,
‘‘message’’ : ‘‘Registro correcto.’’,
‘‘status’’ : ‘‘OK’’
}
```

B.1.2. Logout

Cierra una sesión activa en servidor para un usuario dado.

- **Servlet:**
  `LogoutServlet.java`
- **URL:**
  <servidor:puerto>/TurismoAPI/logout.do
- **Método:**
  `GET`
- **Parámetros:**
  - Sin parámetros
- **Resultado:**
  - *mensaje:* mensaje informativo ofrecido como respuesta. Especialmente útil en caso de error.
  - *status:* estado de la solicitud. "OK" si la petición se ha completado correctamente o "ERROR" si ha ocurrido algún error.
- **Ejemplo:**
  - *Conexión a:* <servidor:puerto>/TurismoAPI/logout.do
  - *Resultado:*

```
{
‘‘message’’ : ‘‘Sesión cerrada correctamente.’’,
‘‘status’’ : ‘‘OK’’
}
```

B.1.3. Buscar usuario

Busca un usuario en el sistema dado su identificador y devuelve información básica sobre el mismo.

- **Servlet:**
  `BuscarUsuServlet.java`
- **URL:**
  <servidor:puerto>/TurismoAPI/buscarusu.do
- **Método:**
  `GET`
- **Parámetros:**
  - *username:* identificador del usuario a buscar.
- **Resultado:**
  - *id:* solo en caso de éxito. Identificador único del usuario.
  - *nombre:* solo en caso de éxito. Nombre completo del usuario.
  - *mensaje:* mensaje informativo ofrecido como respuesta. Especialmente útil en caso de error.
  - *status:* estado de la solicitud. “OK” si la petición se ha completado correctamente, “ERROR” si ha ocurrido algún error y “ESESION” si el usuario peticionario no tiene una sesión activa.
- **Ejemplo:**
  - *Conexión a:*
    <servidor:puerto>/TurismoAPI/buscarusu.do?username=searchuser
  - *Resultado:*

```
{
‘‘id’’ : ‘‘09876’’,
‘‘nombre’’ : ‘‘nombre usuario’’,
‘‘status’’ : ‘‘OK’’
}
```

B.1.4. Modificar datos

Modifica los datos del usuario en el servidor. Solo se envían aquellos datos a modificar, los demás no se alteran.

- **Servlet:**
  `ModificarServlet.java`
- **URL:**
  <servidor:puerto>/TurismoAPI/modificar.do
- **Método:**
  `GET`
- **Parámetros:**
  - *username:* identificador de acceso a la red social a modificar.
  - *nombre:* nuevo nombre completo del usuario.
  - *sexo:* sexo del usuario ("h"/"m").
  - *fecha:* nueva fecha de nacimiento.
  - *latitud:* latitud de la posición actual del usuario.
  - *longitud:* longitud de la posición actual del usuario.
- **Resultado:**
  - *mensaje:* mensaje informativo ofrecido como respuesta. Especialmente útil en caso de error.
  - *status:* estado de la solicitud. "OK" si la petición se ha completado correctamente, "ERROR" si ha ocurrido algún error y "ESESION" si el usuario peticionario no tiene una sesión activa.
- **Ejemplo:**
  - *Conexión a:*
    <servidor:puerto>/TurismoAPI/modificar.do?username=newiden&sexo=m
  - *Resultado:*

```
{
‘‘message’’ : ‘‘Datos modificados correctamente.’’,
‘‘status’’ : ‘‘OK’’
}
```

B.1.5. Recuperar perfil personal

Devuelve todos los datos del perfil personal del usuario (nombre, sexo, etc.).

- **Servlet:**
  `PerfilServlet.java`
- **URL:**
  <servidor:puerto>/TurismoAPI/perfil.do
- **Método:**
  `GET`
- **Parámetros:**
  - Sin parámetros
- **Resultado:**
  - *id:* solo en caso de éxito. Identificador único del usuario dentro de Madrid Live.
  - *rs_ id:* solo en caso de éxito. Identificador único del usuario dentro de la red social a la que se conecte Madrid Live.
  - *username:* solo en caso de éxito. Identificador con el que se ha iniciado sesión en la red social.
  - *nombre:* solo en caso de éxito. Nombre completo del usuario.
  - *sexo:* solo en caso de éxito. Sexo del usuario.
  - *fecha:* solo en caso de éxito. Fecha de nacimiento del usuario.
  - *status:* estado de la solicitud. "OK" si la petición se ha completado correctamente, "ERROR" si ha ocurrido algún error y "ESESION" si el usuario peticionario no tiene una sesión activa.
- **Ejemplo:**
  - *Conexión a:*
    <servidor:puerto>/TurismoAPI/perfil.do
  - *Resultado:*

```
{
‘‘id’’ : ‘‘ML1234’’,
‘‘rs\_id’’ : ‘‘RS1234’’,
‘‘username’’ : ‘‘identificador login’’,
‘‘nombre’’ : ‘‘nombre completo’’,
‘‘sexo’’ : ‘‘h’’,
‘‘fecha’’ : ‘‘11-05-1987’’,
‘‘status’’ : ‘‘OK’’
}
```

B.1.6. Buscar amigos en Madrid Live

Devuelve una lista con todos los amigos del usuario que se encuentran registrados en Madrid Live.

- **Servlet:**
  `BusquedaAmigosServlet.java`
- **URL:**
  <servidor:puerto>/TurismoAPI/buscaramigos.do
- **Método:**
  `POST`
- **Parámetros:**
  - *amigos:* lista de los identificadores únicos en la red social de los amigos separados por comas.
- **Resultado:**
  - *amigos:* solo en caso de éxito. Lista de amigos encontrados junto con su información.
  - *mensaje:* mensaje informativo ofrecido como respuesta. Especialmente útil en caso de error.
  - *status:* estado de la solicitud. "OK" si la petición se ha completado correctamente, "ERROR" si ha ocurrido algún error y "ESESION" si el usuario peticionario no tiene una sesión activa.
- **Ejemplo:**
  - *Conexión a:*
    <servidor:puerto>/TurismoAPI/buscaramigos.do
  - *Resultado:*

```
{
‘‘amigos’’ : [
{
‘‘id’’ : ‘‘ID1’’,
‘‘nombre’’ : ‘‘nombre1’’,
‘‘latitud’’ : ‘‘22.45’’...
},
{
‘‘id’’ : ‘‘ID2’’,
‘‘nombre’’ : ‘‘nombre2’’,
‘‘latitud’’ : ‘‘13.56’...
```

```
}
],
‘‘status’’ : ‘‘OK’’
}
```

B.1.7. Crear un grupo

Recibe los datos del nuevo grupo a crear, lo inserta en la base de datos y devuelve la respuesta al cliente.

- **Servlet:**
  `CCGrupoServlet.java`
- **URL:**
  <servidor:puerto>/TurismoAPI/ccgrupo.do
- **Método:**
  `POST`
- **Parámetros:**
  - *nombre:* nombre para el nuevo grupo.
  - *miembros:* lista de identificadores únicos de los miembros separados por comas.
- **Resultado:**
  - *grupos:* solo en caso de éxito. Lista de todos los grupos creados por este usuario.
  - *mensaje:* mensaje informativo ofrecido como respuesta. Especialmente útil en caso de error.
  - *status:* estado de la solicitud. “OK” si la petición se ha completado correctamente, “ERROR” si ha ocurrido algún error y “ESESION” si el usuario peticionario no tiene una sesión activa.
- **Ejemplo:**
  - *Conexión a:*
    <servidor:puerto>/TurismoAPI/ccgrupo.do
  - *Resultado:*

```
{
‘‘grupos’’ : [
{
‘‘id’’ : ‘‘ID1’’,
‘‘idCreador’’ : ‘‘ID creador 1’’,
```

```
‘‘nombre’’ : ‘‘nombre1’’,
‘‘miembros’’ : [
{
‘‘idMiembro’’ : ‘‘ID miembro 1’’,
‘‘email’’ : ‘‘email miembro 1’’
},
{
‘‘idMiembro’’ : ‘‘ID miembro 2’’,
‘‘email’’ : ‘‘email miembro 2’’
}
, ...
]
},
{
‘‘id’’ : ‘‘ID2’’,
‘‘idCreador’’ : ‘‘ID creador 2’’,
‘‘nombre’’ : ‘‘nombre2’’,
‘‘miembros’’ : [
{
‘‘idMiembro’’ : ‘‘ID miembro 21’’,
‘‘email’’ : ‘‘email miembro 21’’
},
{
‘‘idMiembro’’ : ‘‘ID miembro 22’’,
‘‘email’’ : ‘‘email miembro 22’’
}
, ...
]
}
],
‘‘status’’ : ‘‘OK’’
}
```

B.1.8. Consultar grupos

Devuelve información sobre todos los grupos creados por el usuario.

- **Servlet:**
  `CCGrupoServlet.java`
- **URL:**
  <servidor:puerto>/TurismoAPI/ccgrupo.do
- **Método:**
  `GET`

- **Parámetros:**
  - Sin parámetros
- **Resultado:**
  - *mensaje:* mensaje informativo ofrecido como respuesta. Especialmente útil en caso de error.
  - *status:* estado de la solicitud. "OK" si la petición se ha completado correctamente, "ERROR" si ha ocurrido algún error y "ESESION" si el usuario peticionario no tiene una sesión activa.
- **Ejemplo:**
  - *Conexión a:* <servidor:puerto>/TurismoAPI/ccgrupo.do
  - *Resultado:*

```
{
‘‘grupos’’ : [
{
‘‘id’’ : ‘‘ID1’’,
‘‘idCreador’’ : ‘‘ID creador 1’’,
‘‘nombre’’ : ‘‘nombre1’’,
‘‘miembros’’ : [
{
‘‘idMiembro’’ : ‘‘ID miembro 1’’,
‘‘email’’ : ‘‘email miembro 1’’
},
{
‘‘idMiembro’’ : ‘‘ID miembro 2’’,
‘‘email’’ : ‘‘email miembro 2’’
}
, ...
]
},
{
‘‘id’’ : ‘‘ID2’’,
‘‘idCreador’’ : ‘‘ID creador 2’’,
‘‘nombre’’ : ‘‘nombre2’’,
‘‘miembros’’ : [
{
‘‘idMiembro’’ : ‘‘ID miembro 21’’,
‘‘email’’ : ‘‘email miembro 21’’
},
```

```
{
‘‘idMiembro’’ : ‘‘ID miembro 22’’,
‘‘email’’ : ‘‘email miembro 22’’
}
, ...
]
}
],
‘‘status’’ : ‘‘OK’’
}
```

B.1.9. Borrar grupo

Borra uno de los grupos creados por el usuario.

- **Servlet:**
  `BorrarGrupoServlet.java`
- **URL:**
  <servidor:puerto>/TurismoAPI/borrargrupo.do
- **Método:**
  `GET`
- **Parámetros:**
  - *idGrupo:* identificador único del grupo a eliminar.
- **Resultado:**
  - *mensaje:* mensaje informativo ofrecido como respuesta. Especialmente útil en caso de error.
  - *status:* estado de la solicitud. “OK” si la petición se ha completado correctamente, “ERROR” si ha ocurrido algún error y “ESESION” si el usuario peticionario no tiene una sesión activa.
- **Ejemplo:**
  - *Conexión a:*
    <servidor:puerto>/TurismoAPI/borrargrupo.do?idGrupo=grupo1
  - *Resultado:*

```
{
‘‘mensaje’’ : ‘‘Grupo eliminado correctamente’’,
‘‘status’’ : ‘‘OK’’
}
```

B.2. Gestión del perfil de preferencias

B.2.1. Actualizar el perfil de preferencias (tests)

Actualiza los datos del perfil de preferencias del usuario en base a los resultados de los tests de preferencias.

- **Servlet:**
  `PreferenciasServlet.java`
- **URL:**
  <servidor:puerto>/TurismoAPI/preferencias.do
- **Método:**
  `GET`
- **Parámetros:**
  - *tipo:* tipo de test a realizar. Puede ser "Museos", "Restaurantes" o "Cines"..
  - *ronda:* ronda actual del test de museos.
  - *seleccionado:* elementos seleccionados en el test de museos (separados por comas) o lista ordenada en restaurantes y cines.
  - *noSeleccionado:* elementos no seleccionados en el test de museos (separados por comas).
- **Resultado:**
  - *sol:* solo en caso de éxito. Vector con los tipos de museos que hay que mostrar en la siguiente ronda.
  - *tamanyo:* solo en caso de éxito. Tamaño del vector solución.
  - *ronda_siguiente:* solo en caso de éxito. Número de la próxima ronda del test.
  - *mensaje:* mensaje informativo ofrecido como respuesta. Especialmente útil en caso de error.
  - *status:* estado de la solicitud. "OK" si la petición se ha completado correctamente, "ERROR" si ha ocurrido algún error y "ESESION" si el usuario peticionario no tiene una sesión activa.
- **Ejemplo:**
  - *Conexión a:*
    <servidor:puerto>/TurismoAPI/preferencias.do?tipo=Museos&ronda=2
    &seleccionado=CIENCIAS,PINTURA,HISTORIA
    &noSeleccionado=DEPORTIVO,ARTES_ESCRITAS,INFANTIL

- *Resultado:*

```
{
‘‘sol’’ : [‘‘CIENCIAS’’, ‘‘HISTORIA’’,...],
‘‘tamanyo’’ : 6,
‘‘ronda_siguiente’’ : 3,
‘‘status’’ : ‘‘OK’’
}
```

B.2.2. Sincronizar datos de la red social

Sincroniza los datos de la red social recibidos desde el cliente con el servidor.

- **Servlet:**
  `SincronizacionServlet.java`
- **URL:**
  <servidor:puerto>/TurismoAPI/sincronizar.do
- **Método:**
  `POST`
- **Parámetros:**
  - *tipo:* tipo de datos a sincronizar (amigos, fotos, etc.).
  - *datos:* lista JSON que contiene todos los datos a introducir para el tipo dado.
- **Resultado:**
  - *mensaje:* mensaje informativo ofrecido como respuesta. Especialmente útil en caso de error.
  - *status:* estado de la solicitud. “OK” si la petición se ha completado correctamente, “ERROR” si ha ocurrido algún error y “ESESION” si el usuario peticionario no tiene una sesión activa.
- **Ejemplo:**
  - *Conexión a:*
    <servidor:puerto>/TurismoAPI/sincronizar.do
  - *Resultado:*

```
{
‘‘status’’ : ‘‘OK’’
}
```

B.2.3. Actualizar perfil de preferencias generales

Actualiza el perfil de preferencias generales del usuario (tipos de actividades, disponibilidad de movilidad y horario). Los parámetros recibidos son opcionales y pueden tomar los valores "inc" para incrementar, "dec" para decrementar y "none" para no realizar cambios.

- **Servlet:**
  `PerfilUsuarioServlet.java`
- **URL:**
  <servidor:puerto>/TurismoAPI/perfilusuario.do
- **Método:**
  `POST`
- **Parámetros:**
  - *horario:* cambio en la flexibilidad de horarios del usuario.
  - *movilidad:* cambio en la disponibilidad del usuario a desplazamientos.
  - *parque:* modificación del gusto del usuario por los parques.
  - *paseo:* modificación del gusto del usuario por los paseos.
  - *restaurante:* modificación del gusto del usuario por los restaurantes.
  - *museo:* modificación del gusto del usuario por los museos.
  - *cine:* modificación del gusto del usuario por los cines.
- **Resultado:**
  - *mensaje:* mensaje informativo ofrecido como respuesta. Especialmente útil en caso de error.
  - *status:* estado de la solicitud. "OK" si la petición se ha completado correctamente, "ERROR" si ha ocurrido algún error y "ESESION" si el usuario peticionario no tiene una sesión activa.
- **Ejemplo:**
  - *Conexión a:*
    <servidor:puerto>/TurismoAPI/perfilusuario.do
  - *Resultado:*

```
{
‘‘mensaje’’ : ‘‘Datos correctamente actualizados.’’,
‘‘status’’ : ‘‘OK’’
}
```

B.3. Gestión de recomendaciones y actividades

B.3.1. Recomendaciones generales

Recupera una lista del servidor con las actividades mejor valoradas y los lugares más visitados.

- **Servlet:**
  `LateralServlet.java`
- **URL:**
  <servidor:puerto>/TurismoAPI/lateral.do
- **Método:**
  `GET`
- **Parámetros:**
  - Sin parámetros
- **Resultado:**
  - *mensaje:* mensaje informativo ofrecido como respuesta. Especialmente útil en caso de error.
  - *status:* estado de la solicitud. "OK" si la petición se ha completado correctamente o "ERROR" si ha ocurrido algún error.
- **Ejemplo:**
  - *Conexión a:*
    <servidor:puerto>/TurismoAPI/lateral.do
  - *Resultado:*

```
{
‘‘valorado’’ : ‘‘actividad mas valorada.’’,
‘‘visitado’’ : [‘‘lugar1’’, ‘‘´lugar2’’]
‘‘tamanyo’’ : 2,
‘‘status’’ : ‘‘OK’’
}
```

B.3.2. Recuperar actividades sin valorar

Devuelve una lista al cliente con todas las actividades realizadas por el usuario pendientes de valoración.

- **Servlet:**
  `SinValorarServlet.java`

- **URL:**
  <servidor:puerto>/TurismoAPI/sinvalorar.do
- **Método:**
  `GET`
- **Parámetros:**
  - Sin parámetros
- **Resultado:**
  - *sol:* solo en caso de éxito. Array JSON que contiene la información de todas las actividades pendientes de valoración.
  - *tamanyo:* solo en caso de éxito. Contiene el número de elementos de "sol".
  - *mensaje:* mensaje informativo ofrecido como respuesta. Especialmente útil en caso de error.
  - *status:* estado de la solicitud. "OK" si la petición se ha completado correctamente, "ERROR" si ha ocurrido algún error y "ESESION" si el usuario peticionario no tiene una sesión activa.
- **Ejemplo:**
  - *Conexión a:*
    <servidor:puerto>/TurismoAPI/sinvalorar.do
  - *Resultado:*

```
{
‘‘sol’’ : [
{
‘‘tipo’’ : ‘‘MUSEO’’,
‘‘actividad’’ : {
‘‘nombre’’ : ‘‘Museo Taurino’’,
‘‘direccion’’ : ‘‘Calle de Alcalá 237, Madrid, España",
‘‘resumen’’ : ‘‘’’,
‘‘imagen’’ : ‘‘http://www.a2lenguas.com/images/words/...’’,
‘‘web’’ : ‘‘’’
}
}
],
‘‘tamanyo’’ : ‘‘1’’,
‘‘status’’ : ‘‘OK’’
}
```

B.3.3. Valorar una actividad

Recibe una valoración para una actividad concreta y actualiza los datos del servidor.

- **Servlet:**
  `ValorarServlet.java`
- **URL:**
  <servidor:puerto>/TurismoAPI/valorar.do
- **Método:**
  `GET`
- **Parámetros:**
  - *tipoActividad:* tipo de actividad que se ha valorado (museo, parque, etc.).
  - *recomendacion:* nombre completo de la actividad que se ha valorado.
  - *valoracion:* valoración que le ha dado el usuario a dicha activad. Toma valores comprendidos entre 0 y 5.
- **Resultado:**
  - *mensaje:* mensaje informativo ofrecido como respuesta. Especialmente útil en caso de error.
  - *status:* estado de la solicitud. "OK" si la petición se ha completado correctamente, "ERROR" si ha ocurrido algún error y "ESESION" si el usuario peticionario no tiene una sesión activa.
- **Ejemplo:**
  - *Conexión a:*
    <servidor:puerto>/TurismoAPI/valorar.do?tipo=MUSEO&recomendacion=Museo del Prado &valoracion=4
  - *Resultado:*

```
{
‘‘mensaje’’ : ‘‘Se ha guardado la valoración correctamente..’’,
‘‘status’’ : ‘‘OK’’
}
```

B.3.4. Solicitar una recomendación

Solicita una recomendación a Madrid Live basada en unas restricciones dadas.

- **Servlet:**
  `RecomendadorServlet.java`
- **URL:**
  <servidor:puerto>/TurismoAPI/recomendador.do
- **Método:**
  `POST`
- **Parámetros:**
  - *tipo:* debe tomar siempre el valor "PlantillaConcreta".
  - *horario:* horario en el que se espera realizar el plan, deben indicarse la hora inicial y la final separadas por una coma.
  - *posición:* posición inicial deseada para el plan, se indicarán la latitud y longitud de la posición separadas por una coma.
  - *actividades:* tipos de actividades que el usuario quiere en el plan (museo, parque, cine,etc). Deberán ir separados por comas.
  - *usuarios:* identificadores de los usuarios que van a participar en el plan separados por comas. El usuario con la sesión activa nunca viene en esta lista.
  - *tipoRecomendacion:* indicará el tipo de función de agregación a emplear en recomendaciones grupales. Puede tomar los valores: "maxima", "media", "minima" o "trustmean".
- **Resultado:**
  - *sol:* solo en caso de éxito. Vector JSON con la lista de actividades del plan resultante y la información de cada una de ellas.
  - *tamanyo:* solo en caso de éxito. Número de actividades del plan devuelto.
  - *mensaje:* mensaje informativo ofrecido como respuesta. Especialmente útil en caso de error.
  - *status:* estado de la solicitud. "OK" si la petición se ha completado correctamente, "ERROR" si ha ocurrido algún error y "ESESION" si el usuario peticionario no tiene una sesión activa.
- **Ejemplo:**

- *Conexión a:*
  <servidor:puerto>/TurismoAPI/recomendador.do
- *Resultado:*


```
{
‘‘id’’ : ‘‘-1’’,
‘‘sol’’ : [
{
‘‘tipo’’ : ‘‘MUSEO’’,
‘‘actividad’’ : {
‘‘nombre’’ : ‘‘Museo Taurino’’,
‘‘direccion’’ : ‘‘Calle de Alcalá 237...’’,
‘‘resumen’’ : ‘‘Ubicado en el interior de la Mon...’’,
‘‘imagen’’ : ‘‘http://www.a2lenguas.com/images/w...’’,
‘‘web’’ : ‘‘ ’’,
‘‘horario’’ : ‘‘12:00-14:00’’,
‘‘posicion’’ : {‘‘latitud’’ : ‘‘40.43’’,‘‘longitud’’ : ‘‘-3.66’’},
‘‘paseo’’ : {
‘‘pasos’’ : [],
‘‘tamanyo’’ : ‘‘0’’
},
‘‘tipoRestaurante’’ : ‘‘ ’’
}
},{
‘‘tipo’’ : ‘‘RESTAURANTE’’,
‘‘actividad’’ : {
‘‘nombre’’ : ‘‘La Trucha’’,
‘‘direccion’’ : ‘‘Calle de Gil de Santivañés 4 - 6 (ho...)’’,
‘‘resumen’’ : ‘‘Carne’’,
‘‘imagen’’ : ‘‘http://11870.com/multimedia/imagenes/so_...’’,
‘‘web’’ : ‘‘http://11870.com/pro/la-trucha’’,
‘‘horario’’ : ‘‘14:00-15:30’’,
‘‘posicion’’ : {‘‘latitud’’ : ‘‘40.42’’,‘‘longitud’’ : ‘‘-3.68’’},
‘‘paseo’’ : {
‘‘pasos’’ : [],
‘‘tamanyo’’ : ‘‘0’’
},
‘‘tipoRestaurante’’ : ‘‘Carne’’
}
}
],
‘‘tamanyo’’ : ‘‘2’’,
‘‘usuarios’’ : [‘‘4’’],
‘‘status’’ : ‘‘OK’’
```

```
}
}
```

B.3.5. Aceptar una recomendación

Acepta un plan recomendado y agrega las actividades realizadas a la lista del usuario con la sesión activa.

- **Servlet:**
  `HistorialServlet.java`
- **URL:**
  <servidor:puerto>/TurismoAPI/historial.do
- **Método:**
  `POST`
- **Parámetros:**
  - *plantilla:* JSON que contiene el plan completo realizado por el usuario. Es exactamente el mismo objeto JSON devuelto por la conexión anterior.
- **Resultado:**
  - *mensaje:* mensaje informativo ofrecido como respuesta. Especialmente útil en caso de error.
  - *status:* estado de la solicitud. "OK" si la petición se ha completado correctamente, "ERROR" si ha ocurrido algún error y "ESESION" si el usuario peticionario no tiene una sesión activa.
- **Ejemplo:**
  - *Conexión a:*
    <servidor:puerto>/TurismoAPI/historial.do
  - *Resultado:*

```
{
‘‘mensaje’’ : ‘‘El historial se ha actualizado correctamente’’,
‘‘status’’ : ‘‘OK’’
}
```

Apéndice C

Funcionalidad: casos de uso

En este apéndice se exponen las diferentes funcionalidades de Madrid Live desde un punto de vista más técnico que el visto en los capítulos 3 y 4 siguiendo un modelo básico de casos de uso y diagramas de flujo.

C.1. Inicio de sesión

- **Precondición**: el usuario no debe tener una sesión iniciada en el sistema.
- **Postcondición**: se creará una sesión para el usuario y se le dará acceso al sistema.
- **Resumen**: para comenzar a usar la aplicación, el usuario debe iniciar sesión. Este inicio de sesión actualmente está integrado con Facebook, por lo que se deberán usar las credenciales de acceso de dicha plataforma.
- **Flujo Normal**: (Figura C.1)
  - ***Paso 1:*** el usuario accede a la aplicación y el sistema le muestra la pantalla de inicio de sesión.
  - ***Paso 2:*** si dicho usuario ya tiene una sesión iniciada en la aplicación de Facebook de su dispositivo, ir al paso 5.
  - ***Paso 3:*** el usuario introduce sus credenciales de acceso (usuario y contraseña) y selecciona la opción de "iniciar sesión".
  - ***Paso 4:*** se comprueban las credenciales de acceso con la red social (Facebook). En caso de no ser correctas, avisar al usuario e ir al paso 1.

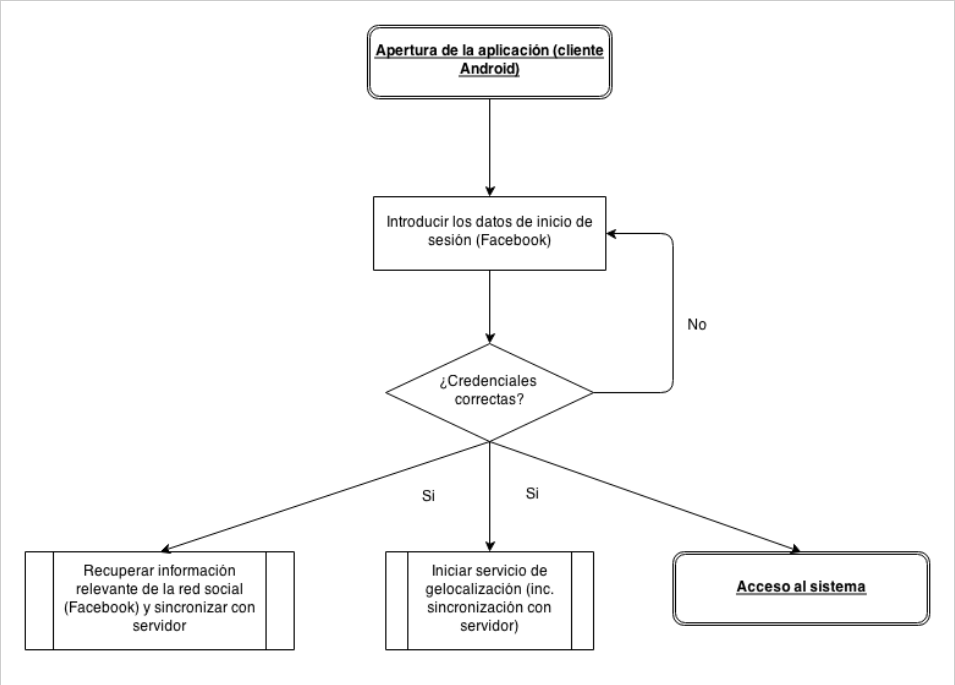


Figura C.1: Diagrama de flujo: inicio de sesión

- ***Paso 5:*** con los datos de acceso correctamente verificados se realizan tres tareas en paralelo: iniciar una sesión para el usuario, comenzar la recuperación de datos de la red social (Facebook) y dar acceso al sistema el usuario.

C.2. Sincronización de datos

- **Precondición**: el usuario debe haber iniciado sesión en el sistema.
- **Postcondición**: se recuperarán y sincronizarán todos los datos relevantes de la cuenta en la red social (Facebook) del usuario.
- **Resumen**: tras un inicio de sesión, y cada 24 horas, se recuperará la información más relevante (amigos, lista de imágenes, etiquetas, etc.) de la red social (Facebook en nuestro caso) y se sincronizarán con el servidor. Estos datos se emplearán para mejorar las recomendaciones.
- **Flujo Normal**: (Figura C.2)
  - ***Paso 1:*** se comprueba si (a) el usuario acaba de iniciar sesión o (b) han pasado al menos 24 horas desde la última sincronización de datos.

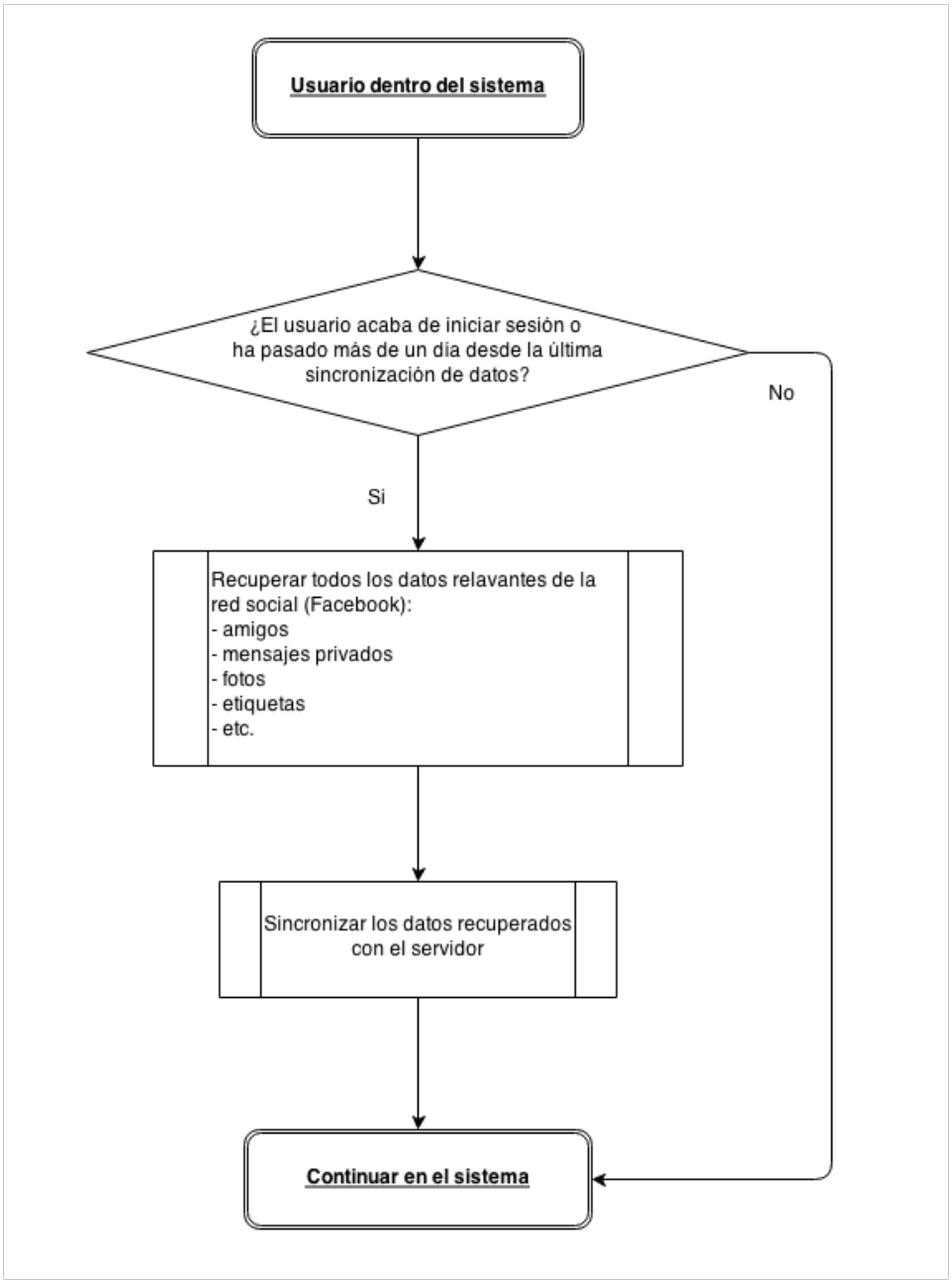


Figura C.2: Diagrama de flujo: sincronización de datos

- ***Paso 2:*** en caso afirmativo, se recuperarán todos los datos necesarios (lista de amigos, fotos, mensajes, etc.) en segundo plano.
- ***Paso 3:*** dichos datos se irán sincronizando con el servidor a medida que se vayan recuperando.

C.3. Recomendaciones iniciales

- **Precondición**: el usuario debe tener sesión iniciada en el sistema.
- **Postcondición**: se le mostrarán al usuario una serie de recomendaciones básicas.
- **Resumen**: en la pantalla principal de la aplicación Android se mostrarán varias recomendaciones basadas en actividades realizadas por otros usuarios. Concretamente, las actividades mejor valoradas y los lugares más visitados.
- **Flujo Normal**: (Figura C.3)
  - ***Paso 1:*** el usuario accede a la pantalla principal de la aplicación.
  - ***Paso 2:*** el sistema solicita, en segundo plano, las actividades que serán mostradas.
  - ***Paso 3:*** en caso de que se hayan recuperado datos de estas actividades, se mostrarán en los apartados correspondientes. De haber ocurrido algún error, estos espacios quedarán vacíos.

C.4. Tests de preferencias

- **Precondición**: el usuario debe tener sesión iniciada en el sistema.
- **Postcondición**: se actualizará el perfil de preferencias del usuario en función de las respuestas a los tests.
- **Resumen**: el usuario tendrá la opción de realizar varios tests que permitirán al Madrid Live generar una versíón más precisa de su perfil de preferencias. Esto contribuirá a mejorar las recomendaciones para dicho usuario.
- **Flujo Normal - test de museos**: el test de museos está basado en el algoritmo ELO (ver sección 3.3.4) y se realiza en 4 rondas.(Figura C.4)
  - ***Paso 1:*** se presentan al usuario 6 tipos de museo aleatorios. Éste debe seleccionar los que más le gusten hasta un máximo de 3 de ellos y pulsar el botón de "Enviar".

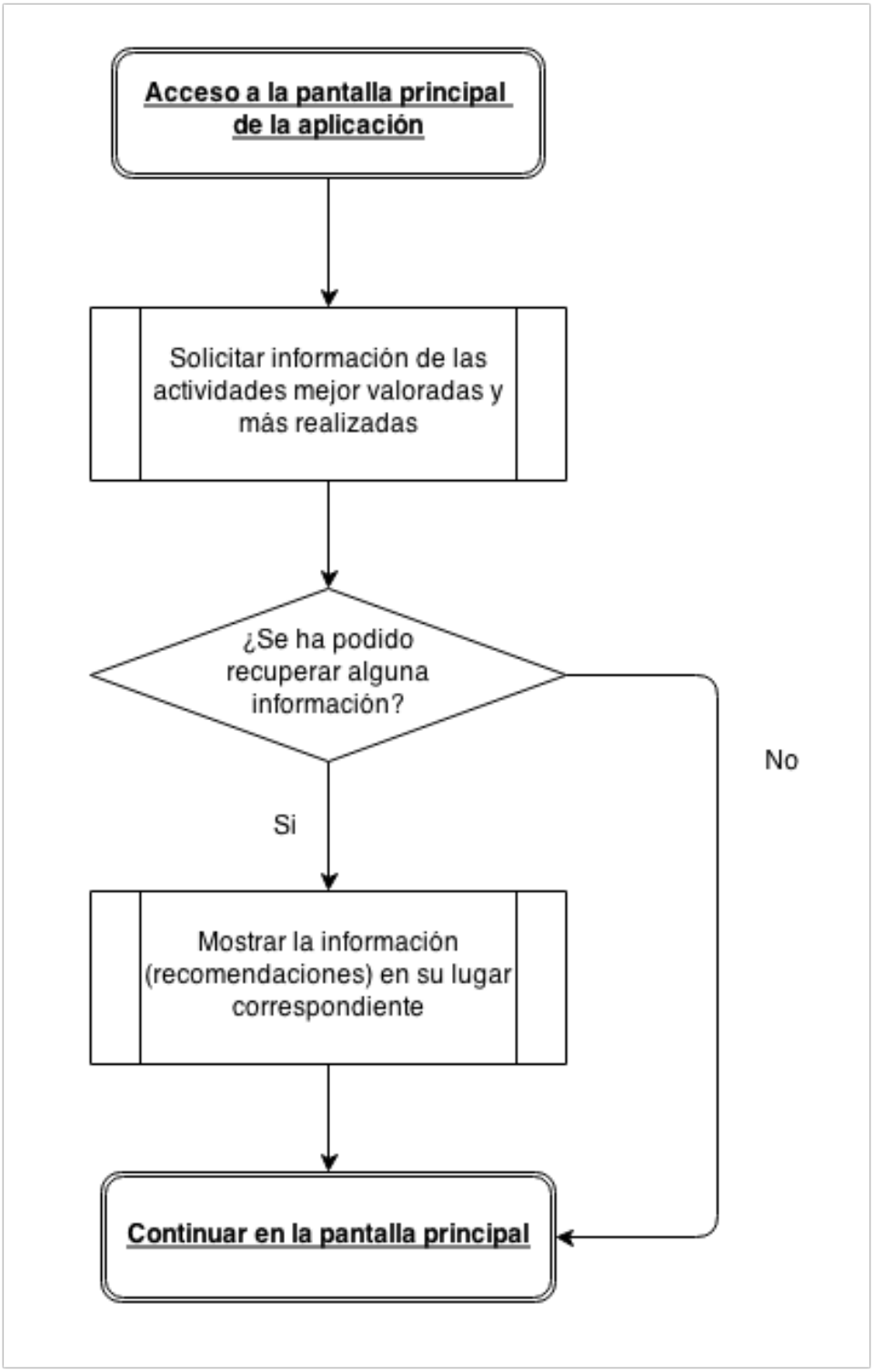


Figura C.3: Diagrama de flujo: recomendaciones iniciales

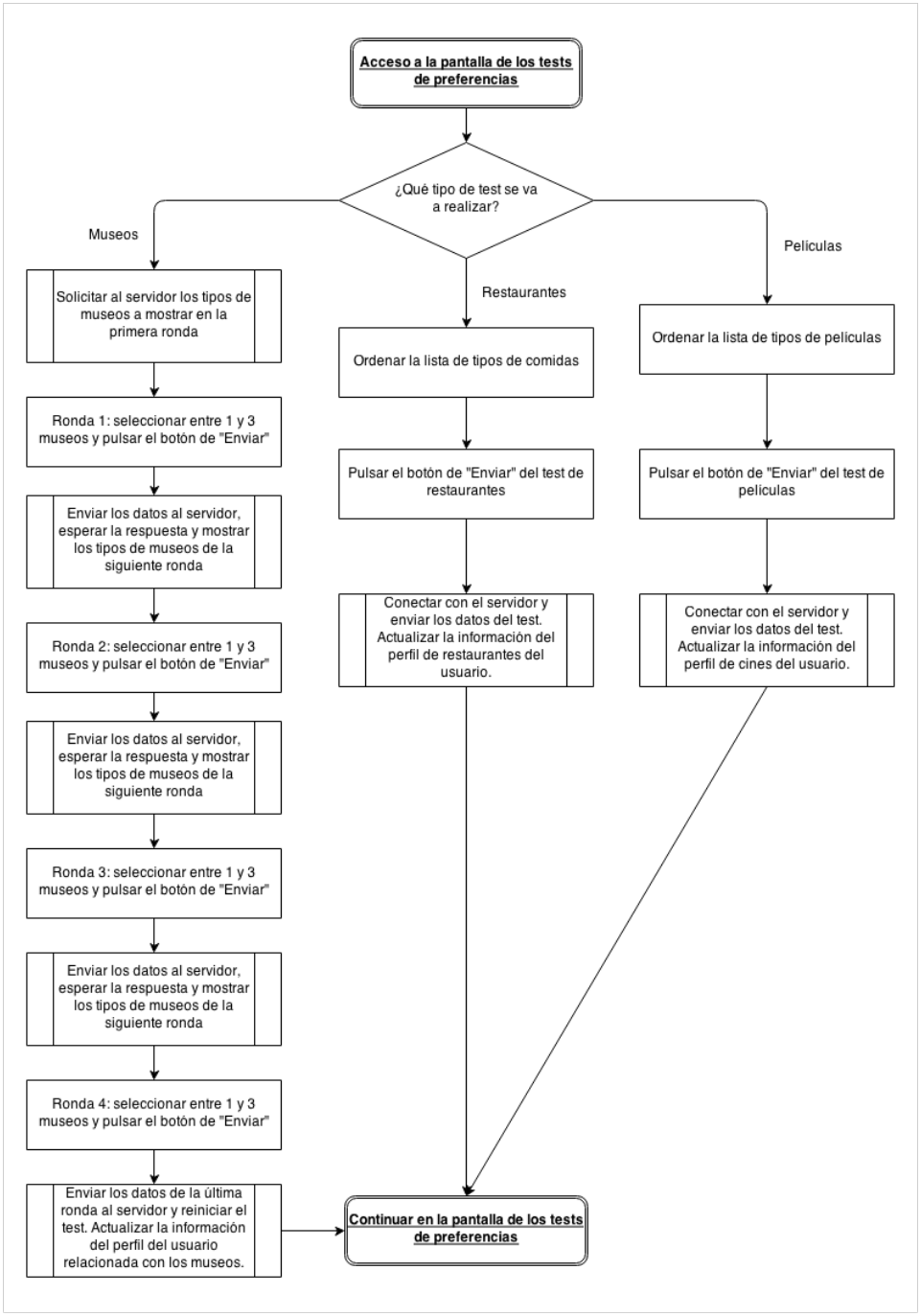


Figura C.4: Diagrama de flujo: tests de preferencias

- ***Paso 2:*** el sistema calcula una nueva ronda de 6 tipos de museo a mostrar en función de las opciones elegidas por el usuario en la ronda 1. 3 de los tipos de museo se corresponderán con los pertenecientes al grupo de elementos más valorados por el usuario y los otros 3 a los segundos más valorados. De nuevo, deben elegirse los tipos de museo preferidos con un máximo de 3 elementos y pulsar el botón de "Enviar".
- ***Paso 3:*** el sistema calcula una nueva ronda de 6 tipos de museo a mostrar en función de las opciones elegidas por el usuario en la ronda 2. En este caso, 3 de los tipos de museo se corresponderán con los pertenecientes al grupo de elementos más valorados mientras que los otros 3 serán los que no han aparecido en la ronda anterior. Como antes, deben seleccionarse los tipos de museo preferidos con un máximo de 3 elementos y pulsar el botón de "Enviar".
- ***Paso 4:*** por último, el sistema calcula de nuevo una nueva ronda de 6 tipos de museo en función de las opciones elegidas por el usuario en las rondas anteriores. En concreto, se mostrarán los elementos con mejores valoraciones. De nuevo, deben elegirse los tipos de museo preferidos con un máximo de 3 elementos y pulsar el botón de "Enviar".
- ***Paso 5:*** el servidor calcula los resultados de todas las rondas y actualiza el perfil de preferencias de museos del usuario.

- **Flujo Normal - test de restaurantes**: (Figura C.4)
  - ***Paso 1:*** se muestra al usuario una lista de tipos de comida que debe ordenar en función de sus gustos.
  - ***Paso 2:*** el usuario ordena todos los elementos de la lista.
  - ***Paso 3:*** el servidor recibe la lista ordenada y actualiza el perfil de preferencias de restaurantes en función de los resultados.
- **Flujo Normal - test de películas**: (Figura C.4)
  - ***Paso 1:*** se muestra al usuario una lista de tipos de películas que debe ordenar en función de sus gustos.
  - ***Paso 2:*** el usuario ordena todos los elementos de la lista.
  - ***Paso 3:*** el servidor recibe la lista ordenada y actualiza el perfil de preferencias de cines en función de los resultados.

C.5. Consulta de datos personales

- **Precondición**: el usuario debe tener sesión iniciada en el sistema.

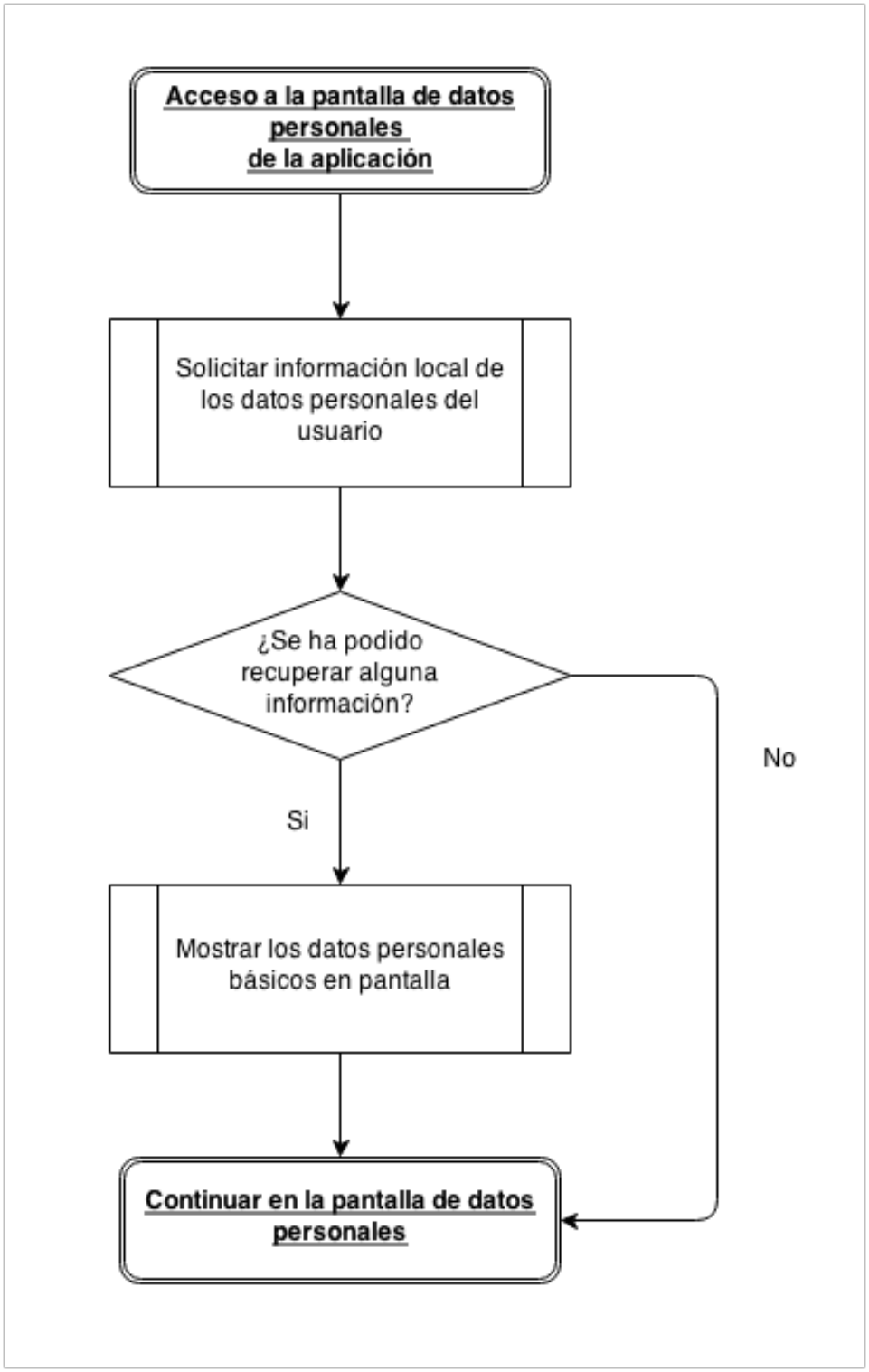


Figura C.5: Diagrama de flujo: consulta de datos personales

- **Postcondición**: se mostrarán al usuario sus datos personales.
- **Resumen**: tras acceder a la pantalla de datos del usuario éste podrá visualizar alguna información básica relacionada con sus datos peronales.
- **Flujo Normal**: (Figura C.5)
  - ***Paso 1:*** el usuario accede a la pantalla de visualización de datos personales.
  - ***Paso 2:*** el sistema trata de recuperar la información solicitada. En concreto: nombre completo, email, fecha de nacimiento y sexo..
  - ***Paso 3:*** si se han recuperado los datos correctamente, se mostrarán en las secciones correspondientes. En caso contrario permanecerán vacías y se notificará al usuario.

C.6. Inicio/detención de la geolocalización

- **Precondición**: el usuario debe haber iniciado sesión en el sistema.
- **Postcondición**: el sistema de geolocalización/posicionamiento se activará o desactivará.
- **Resumen**: una vez dentro del sistema el usuario tendrá la opción de activar o desactivar el servicio de geolocalización.
- **Flujo Normal**: (Figura C.6)
  - ***Paso 1:*** estando en cualquier sección de la aplicación cliente, el usuario despliega el menú principal.
  - ***Paso 2:*** si el servicio de geolocalización está activado se le mostrará la opción de "Detener geolocalización", en caso contrario visualizará la de "Iniciar geolocalización".
  - ***Paso 3:*** el usuario selecciona la opción correspondiente. En función de la deseada (iniciar/detener) se activará o detendrá el servicio de posicionamiento y la sincronización de datos con el servidor.

C.7. Creación de grupos

- **Precondición**: el usuario debe haber iniciado sesión en el sistema.
- **Postcondición**: se creará un nuevo grupo personalizado de usuarios para posteriores recomendaciones.

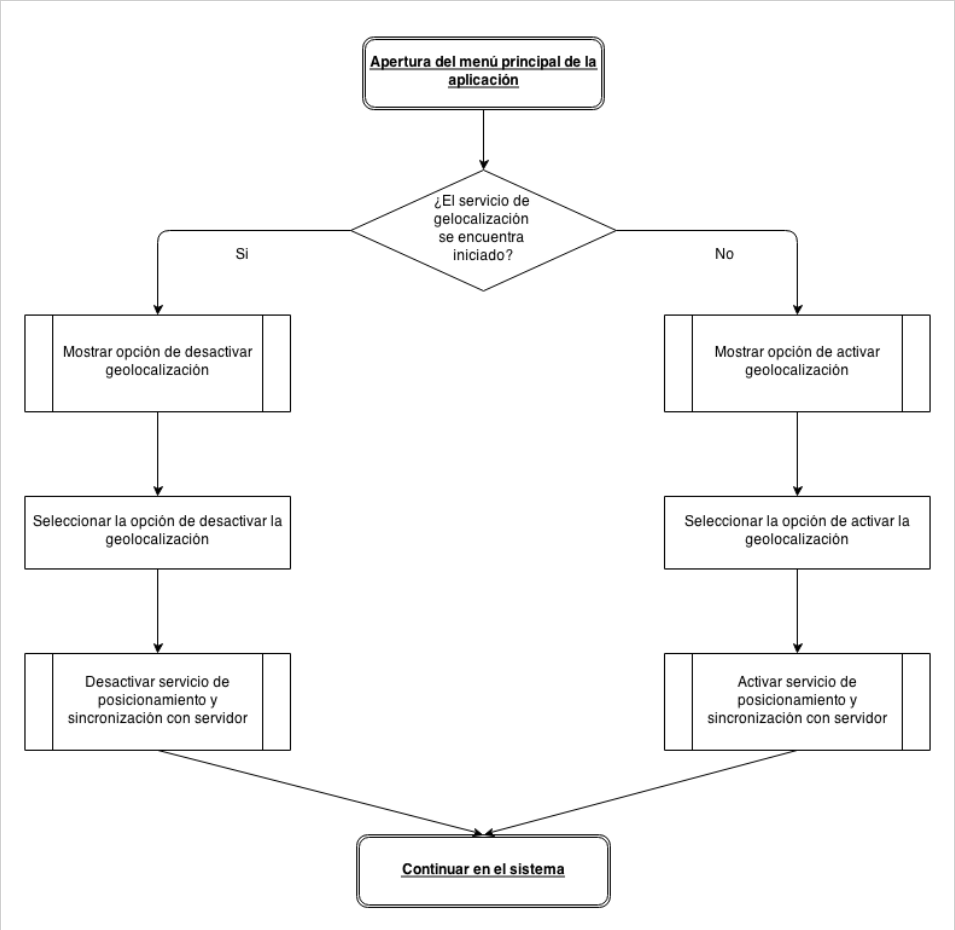


Figura C.6: Diagrama de flujo: activación y desactivación de la geolocalización

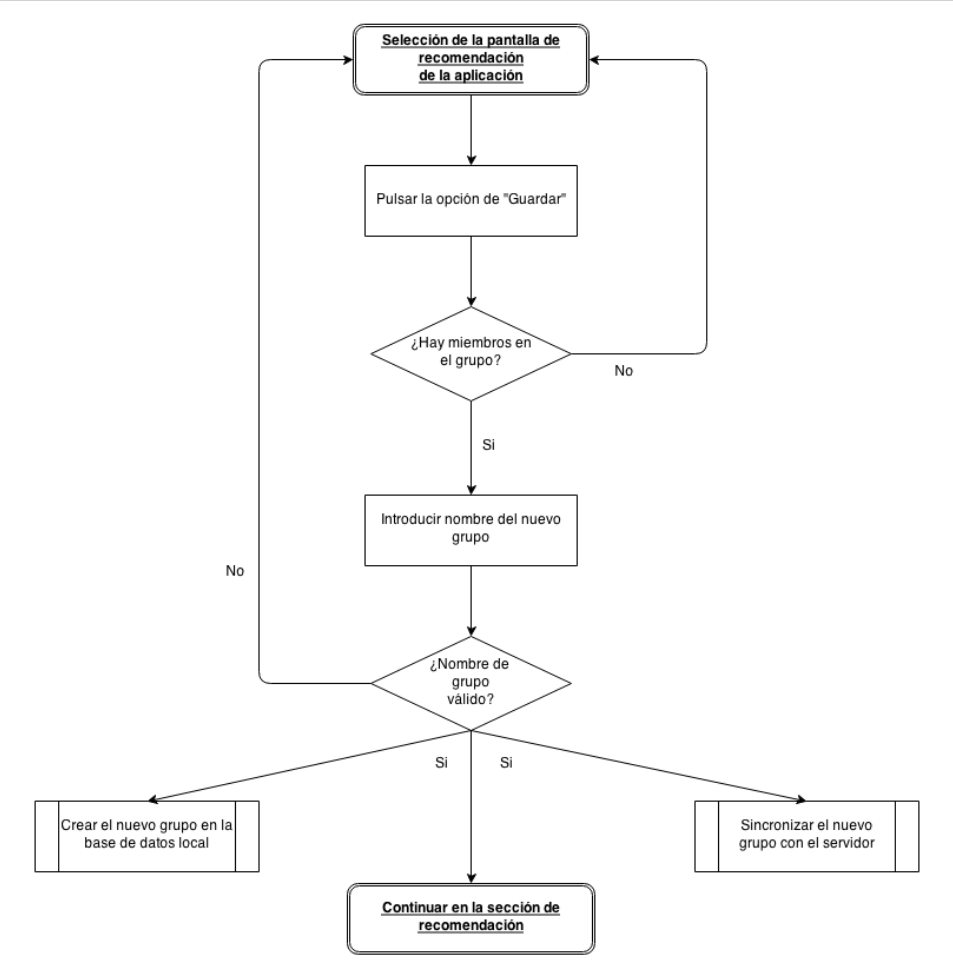


Figura C.7: Diagrama de flujo: creación de grupos

- **Resumen**: desde la pantalla de recomendación el usuario tendrá la opción de crear un nuevo grupo de usuarios.
- **Flujo Normal**: (Figura C.7)
  - ***Paso 1:*** el usuario accede a la sección de recomendación del cliente Android y agrega los amigos deseados al plan. Acto seguido pulsa la opción de "Guardar".
  - ***Paso 2:*** si no hay amigos en el plan (a mayores del usuario original) se avisa de ello y se vuelve al paso inicial.
  - ***Paso 3:*** se solicita mediante un cuadro de diálogo el nombre que se le dará al nuevo grupo.
  - ***Paso 4:*** en caso de no ser un nombre válido, volver al paso inicial. De lo contrario (nombre válido) se mantendrá al usuario en la pantalla de recomendación y, en paralelo, se creará el nuevo grupo en la base de datos local y se sincronizará el cambio con el servidor.

C.8. Carga de un grupo

- **Precondición**: el usuario debe haber iniciado sesión en el sistema y tener algún grupo creado.
- **Postcondición**: se cargará un grupo predefinido y se agregará al plan de recomendación actual.
- **Resumen**: el usuario podrá cargar grupos de amigos a través de una opción disponible en la pantalla de recomendación.
- **Flujo Normal**: (Figura C.8)
  - ***Paso 1:*** el usuario accede a la sección de solicitud de recomendación del cliente Android y pulsa la opción de "Cargar".
  - ***Paso 2:*** se despliega un menú que muestra los grupos disponibles (en caso de existir alguno).
  - ***Paso 3:*** el usuario selecciona el grupo de amigos que desea cargar dentro de todos los de la lista.
  - ***Paso 4:*** se eliminan todos los amigos que había previamente en el plan y se cargan los pertenecientes al grupo seleccionado.

C.9. Borrado de un grupo

- **Precondición**: el usuario debe haber iniciado sesión en el sistema y tener algún grupo creado.

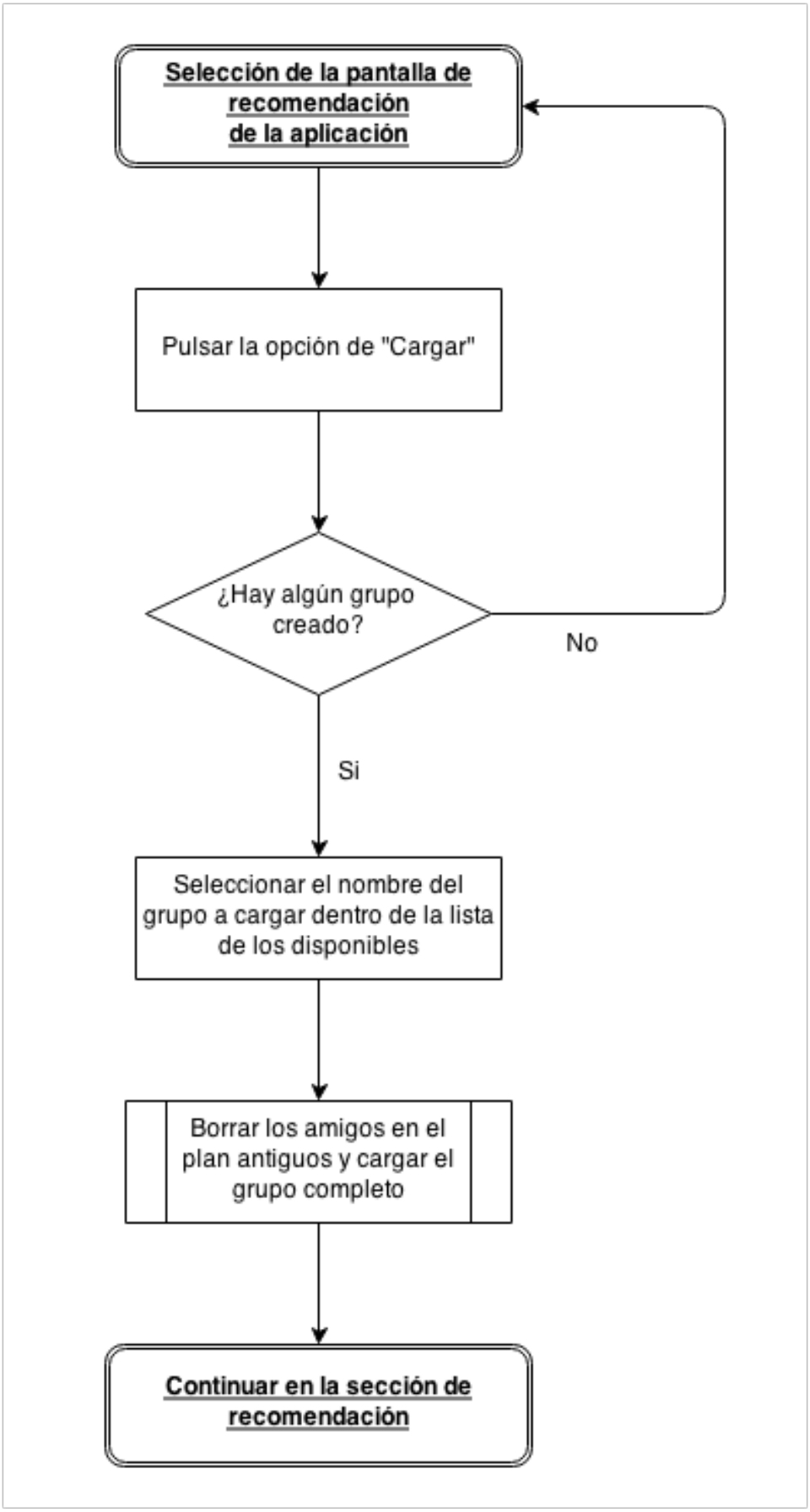


Figura C.8: Diagrama de flujo: carga de grupos

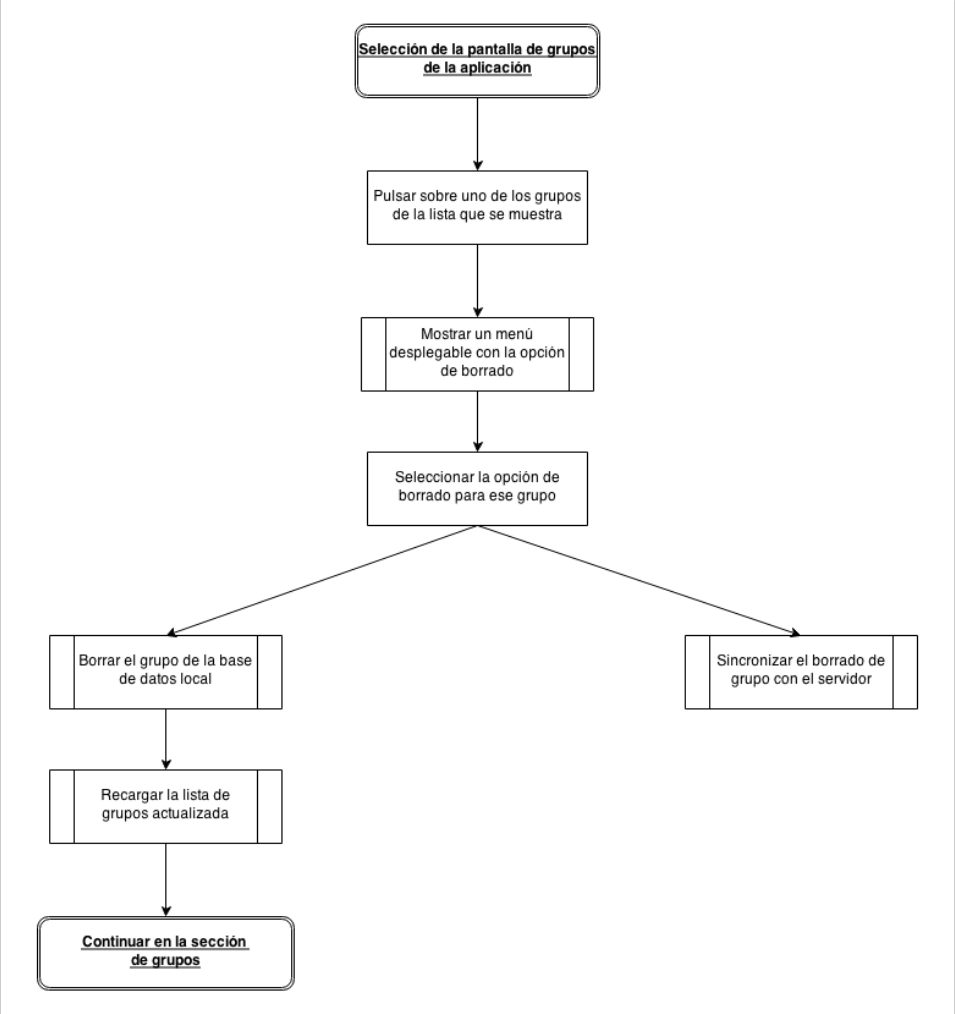


Figura C.9: Diagrama de flujo: borrado de un grupo

- **Postcondición**: se eliminará el grupo deseado dentro de los definidos por el usuario.
- **Resumen**: se podrán eliminar grupos previamente creados a través de la correspondiente sección accesible desde el menú principal.
- **Flujo Normal**: (Figura C.9)
  - ***Paso 1:*** el usuario accede a la pantalla de visualización de los grupos creados a través del menú principal.
  - ***Paso 2:*** en la lista que se muestra, el usuario selecciona el grupo que desea eliminar.
  - ***Paso 3:*** se muestra un cuadro de diálogo con un botón que da la opción de eliminar dicho grupo. El usuario selecciona dicha opción.
  - ***Paso 4.1:*** se elimina el grupo deseado de la base de datos local y se recarga la lista de la pantalla de grupos.
  - ***Paso 4.2:*** en paralelo al paso 4.1 se sincroniza la eliminación del grupo con el servidor.

C.10. Solicitud de recomendación

- **Precondición**: el usuario debe haber iniciado sesión en el sistema.
- **Postcondición**: se solicitará un plan de ocio al sistema.
- **Resumen**: desde la pantalla de recomendación, el usuario tendrá acceso a un pequeño formulario donde podrá configurar los parámetros del plan a realizar a su gusto. Tras esto, podrá solicitar a Madrid Live un plan que encaje con dichas preferencias.
- **Flujo Normal**: (Figura C.10)
  - ***Paso 1:*** el usuario accede a la sección o pantalla de recomendación. En ese momento, el sistema busca los amigos cercanos y agrega los detectados al plan actual.
  - ***Paso 2:*** si el usuario quiere agregar nuevos amigos, introducirá sus emails en el campo de texto y pulsará el botón de "agregar". Si el amigo no es usuario de Madrid Live no se incorporará al plan. También existirá la opción de cargar un grupo completo de amigos en vez de hacerlo de forma individual (explicado anteriormente).
  - ***Paso 3:*** si el usuario quiere un horario diferente al predeterminado (hora actual) deberá hacer uso de la opción "Plan para más tarde" y seleccionar ahí la hora de inicio y final del plan deseado.

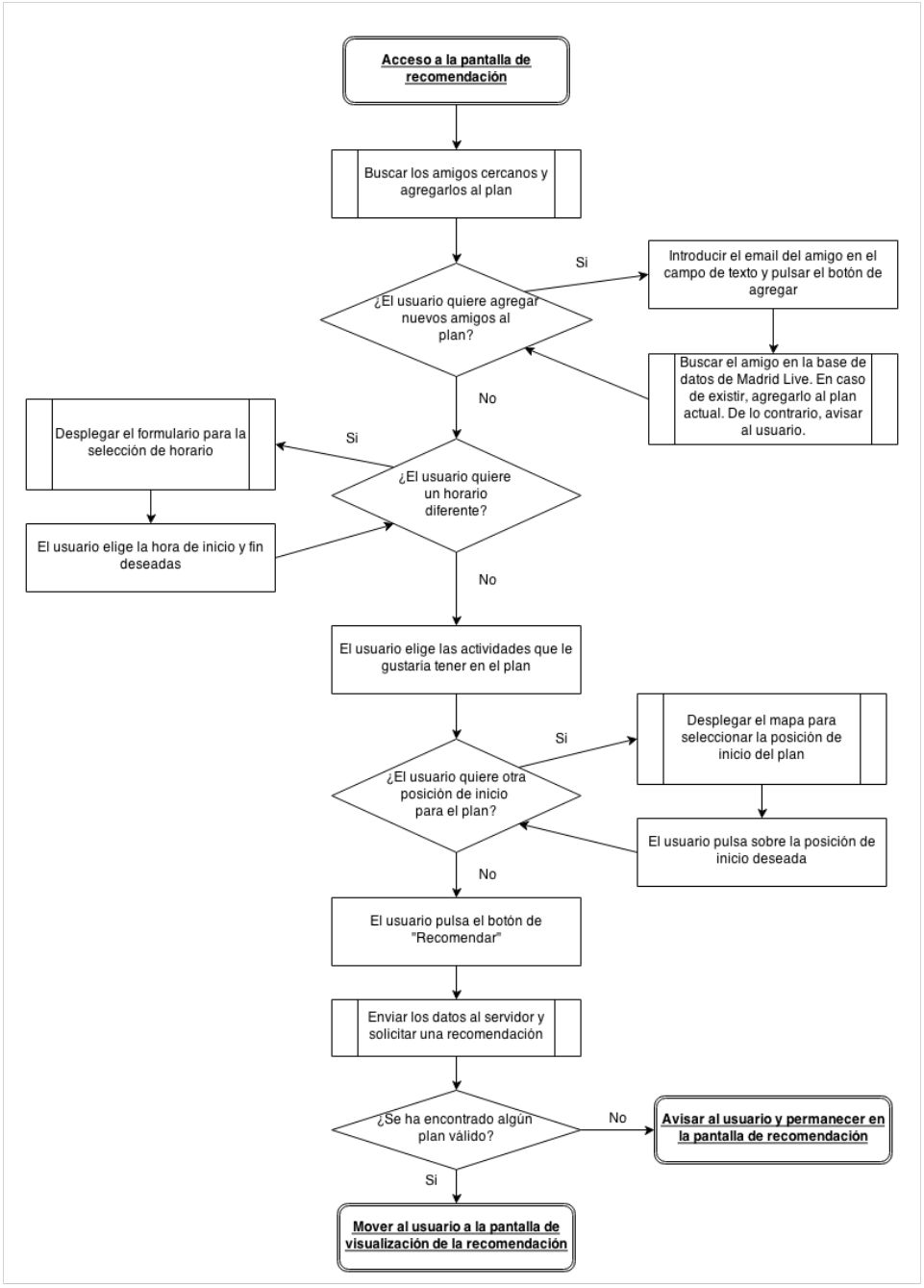


Figura C.10: Diagrama de flujo: solicitud de una recomendación

- ***Paso 4:*** a continuación el usuario debe elegir qué actividades le gustaría que apareciesen en su plan. Puede lograr esto a través de los *checkbox* de la sección de "Actividades".
- ***Paso 5:*** el usuario también puede elegir una posición donde comenzar el plan - por defecto Madrid Live elige la posición actual -. Para ello debe hacer uso de la opción "Elegir posición" del apartado "Lugar". Ahí se desplegará un mapa que le permitirá marcar la posición de inicio.
- ***Paso 6:*** por último, con todos los parámetros completos, se procederá a solicitar la recomendación a través del botón "Recomendar".
- ***Paso 7:*** si el sistema ha podido encontrar un plan de ocio con los parámetros establecidos, éste se mostrará en la pantalla de resultado de la recomendación. En caso contrario, se avisará al usuario para permitirle modificar alguno de los parámetros.

C.11. Resultado de una recomendación

- **Precondición**: el usuario debe tener una sesión activa en el sistema y haber solicitado un plan de ocio a través de la pantalla de recomendación.
- **Postcondición**: se mostrará el plan de ocio propuesto y se dará la opción de aceptarlo o rechazarlo.
- **Resumen**: tras la solicitud de un plan de ocio, se le muestra al usuario en una lista de actividades. Cada una de ellas tendrá su propia información (enlace a Google Maps, dirección, puntos de interés, etc.). Además, se dará la opción de aceptar o rechazar el plan propuesto.
- **Flujo Normal**: (Figura C.11)
  - ***Paso 1:*** se tomará la lista de actividades del plan propuesto que viene de servidor y se mostrará al usuario a través de la pantalla de resultado de la recomendación.
  - ***Paso 2:*** el usuario podrá elegir si acepta o rechaza el plan:
  - ***Paso 2.1:*** en caso de aceptarlo, se notificará esta acción al servidor y se actualizarán su historial de actividades y sus preferencias (horario y desplazamiento).
  - ***Paso 2.1:*** en caso de aceptarlo, se notificará esta acción al servidor y se actualizarán sus preferencias (horario y desplazamiento).
  - ***Paso 3:*** si el módulo de captura de imágenes está activo y el dispositivo tiene cámara frontal, se tomará una imagen del momento

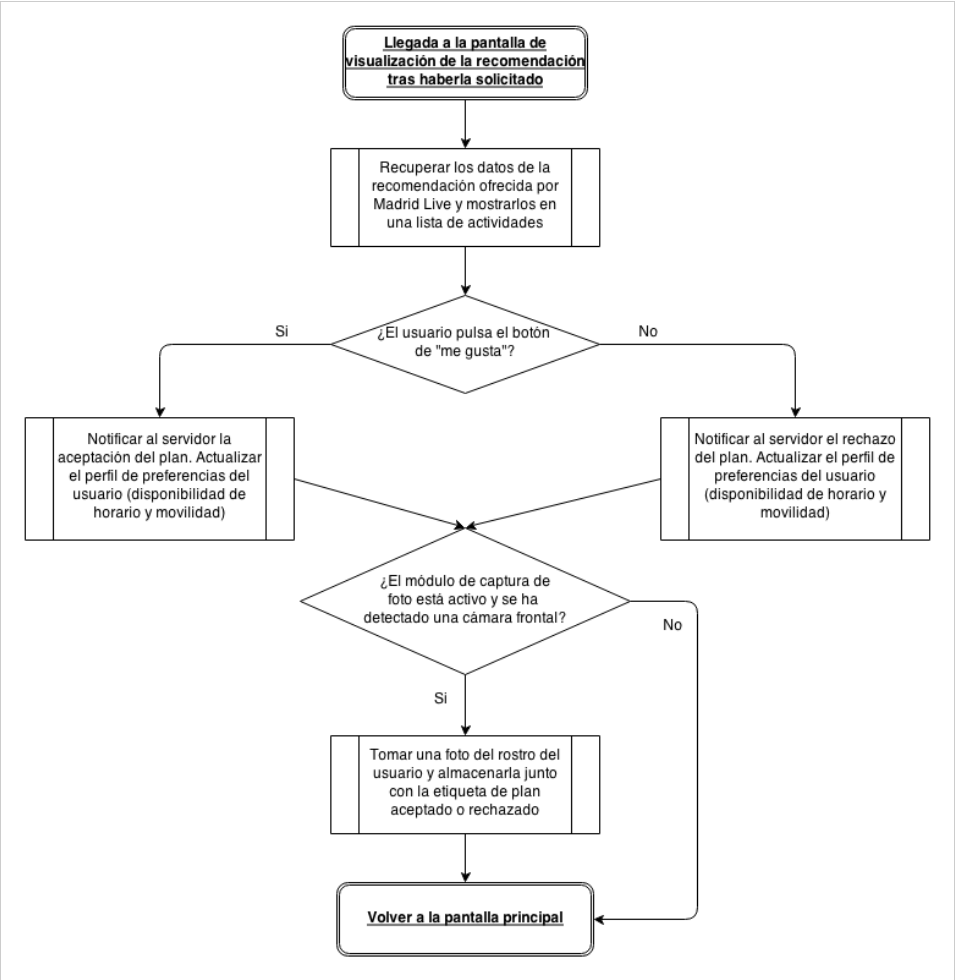


Figura C.11: Diagrama de flujo: resultado de una recomendación

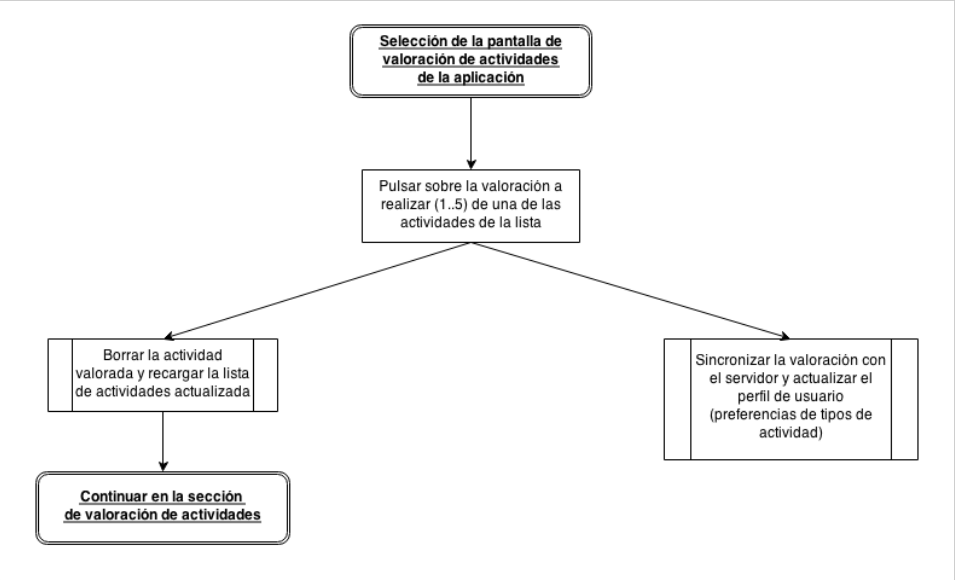


Figura C.12: Diagrama de flujo: valoración de actividades

en el que se acepta/rechaza el plan y se almacenará en un registro local junto con la etiqueta que indica si el plan ha gustado o no al usuario.

- ***Paso 4:*** se redirigirá al usuario a la pantalla principal de la aplicación.

C.12. Valoración de actividades

- **Precondición**: el usuario debe tener una sesión activa en el sistema y haber realizado al menos una actividad.
- **Postcondición**: se podrán valorar las actividades realizadas y se reajustará el perfil de usuario en función de esto.
- **Resumen**: Madrid Live da la opción a los usuarios de valorar las actividades realizadas. Esto permite reajustar el perfil de preferencias del usuario y afinar futuras recomendaciones.
- **Flujo Normal**: (Figura C.12)
  - ***Paso 1:*** el usuario accede a la pantalla de valoración. En ella se carga la lista de actividades realizadas y no valoradas. Cada una de ellas tendrá la opción de ser votada con un valor entre 1 y 5 (siendo 1 que no ha gustado y 5 que ha gustado mucho).
  - ***Paso 2:*** el usuario valora una de las actividades.

- ***Paso 3:*** se notifica la puntuación al servidor y se actualiza el perfil de preferencias del usuario para ese tipo de actividad.
- ***Paso 4:*** se recarga la lista de actividades eliminando la recién valorada. El usuario permanece en la pantalla de valoración.

C.13. Cierre de sesión

- **Precondición**: el usuario debe tener una sesión iniciada en la aplicación.
- **Postcondición**: la sesión activa será finalizada y se sacará al usuario del sistema.
- **Resumen**: el usuario podrá cerrar la sesión activa y eliminar las credenciales en cualquier momento a través del menú principal.
- **Flujo Normal**: (Figura C.13)
  - ***Paso 1:*** desde cualquier sección el usuario despliega el menú principal y selecciona la opción de "Cerrar sesión".
  - ***Paso 2:*** se establece una conexión con el servidor y se solicita el cierre de la sesión remota. En caso de fallo, se notifica al usuario y no se continúa.
  - ***Paso 3:*** se procede a detener el servicio de posicionamiento/-geolocalización.
  - ***Paso 3:*** las credenciales de sesión referentes a la red social (Facebook) son eliminadas.
  - ***Paso 3:*** el usuario es redirigido a la pantalla de inicio de sesión.

C.14. Generación de recomendación individual (servidor)

- **Precondición**: el cliente Android debe haber solicitado una recomendación individual al servidor y enviado los parámetros - restricciones - de la misma.
- **Postcondición**: el servidor devolverá al cliente el plan de ocio que mejor encaje con los parámetros de recomendación.
- **Resumen**: este caso de uso corresponde a la continuación de la solicitud de una recomendación por parte del cliente. El principal objetivo es mostrar, de forma muy básica, el funcionamiento de la recomendación en la parte de *backend*. En un capítulo posterior se detallará el algoritmo de recomendación completo.

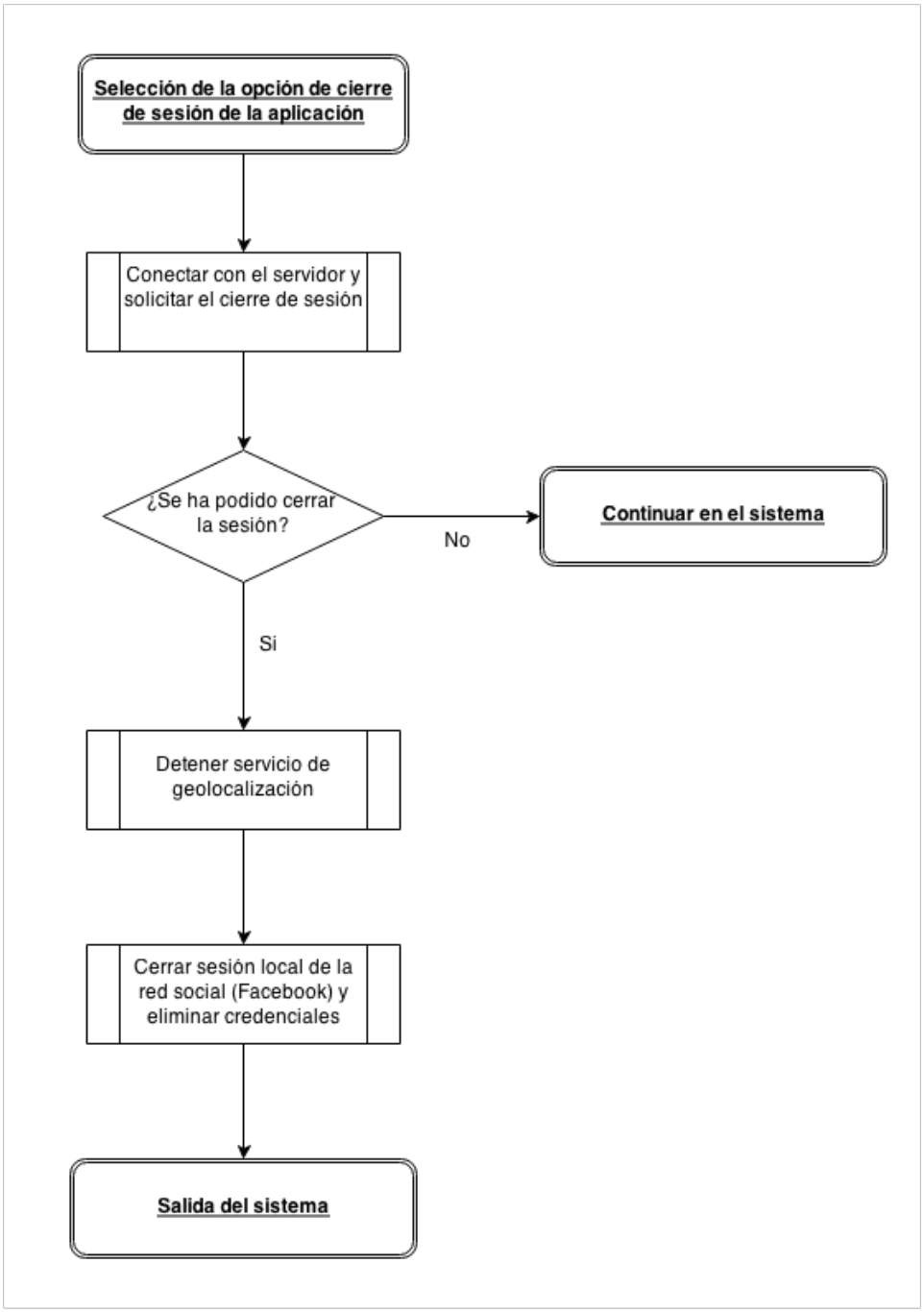


Figura C.13: Diagrama de flujo: cierre de sesión

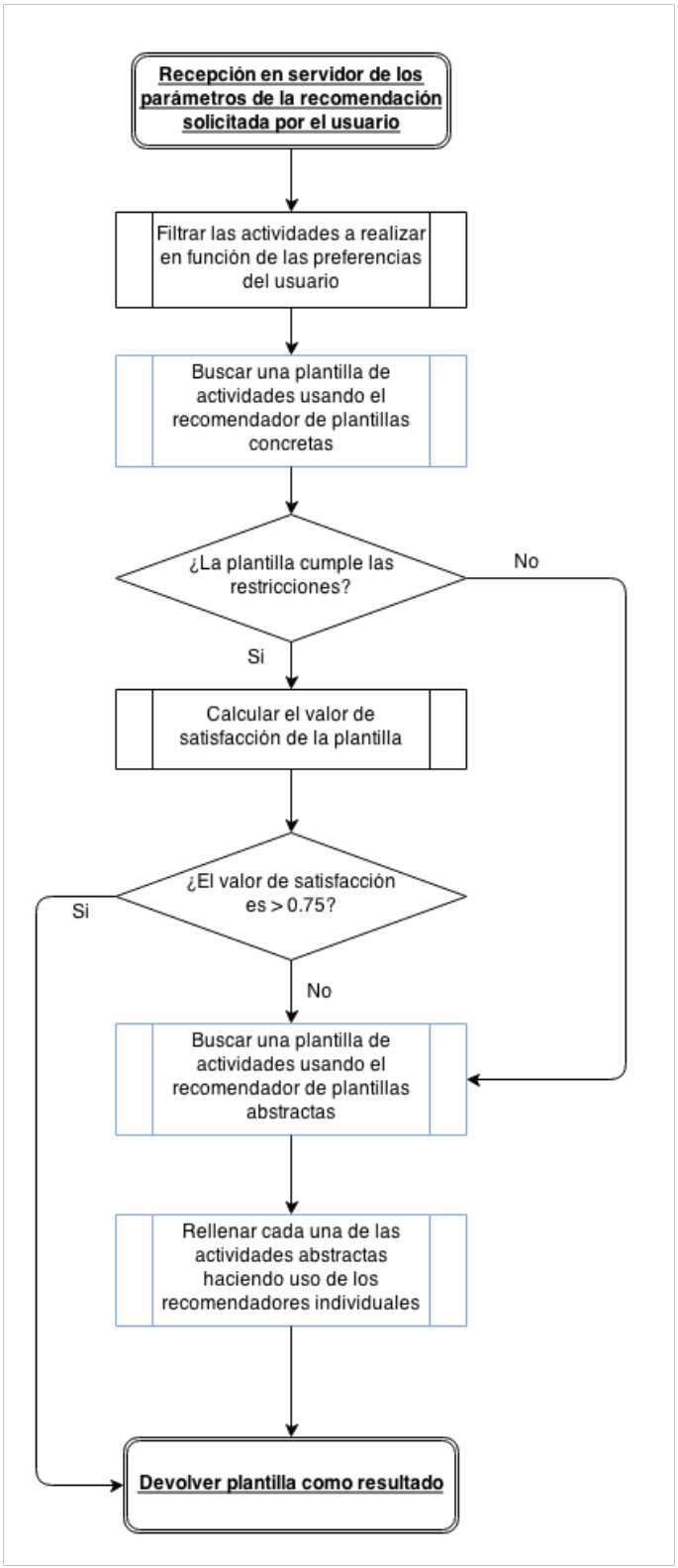


Figura C.14: Diagrama de flujo: generación de una recomendación individual

- **Flujo Normal**: (Figura C.14)

  - ***Paso 1:*** el servidor recibe los parámetros de recomendación del cliente (horario, lugar, actividades, etc.).

  - ***Paso 2:*** se realiza un primer filtrado sobre el número y tipo de actividades a mostrar. Se combinan para ello la predisposición del usuario a realizar planes fuera del horario señalado y sus preferencias por cada tipo de actividad. Por ejemplo: si un usuario da un margen de 2 horas para un plan, marca los 5 tipos de actividad para ser realizados en ese intervalo y, además, a dicho usuario no le gusta que se le recomienden planes fuera del horario previsto; entonces el sistema recortará las 5 actividades solicitadas y dejará solo una de ellas - la preferida del usuario -. Por otro lado, si otro usuario marca un intervalo de 2 horas para su plan con 5 actividades pero a éste no le importa que el plan propuesto se salga del horario, Madrid Live no recortaría ninguna actividad y alargaría el plazo del plan previsto.

  - ***Paso 3:*** a continuación se llama al recomendador de plantillas concretas para buscar planes realizados por otros usuarios. Para cada plan recuperado se comprueba si cumple las restricciones impuestas (horario, lugar, etc.). En caso afirmativo, se calcula un valor de "satisfacción", Éste indicará el nivel de agrado de dicho plan para el usuario.

  - ***Paso 4:*** si se encuentra algún plan entre las plantillas concretas con valor de satisfacción mayor a 0.75[1] entonces se devuelve dicho plan como propuesta para el usuario.

  - ***Paso 5:*** si no se ha encontrado un plan dentro de las plantillas concretas, entonces se llamará al recomendador de plantillas abstractas. Éste se encarga de generar un grupo de actividades no definidas. Por ejemplo: un museo y un parque de 15:00 a 19:00.

  - ***Paso 6:*** acto seguido, para cada actividad abstracta presente en la plantilla anterior, se llamará al correspondiente recomendador de actividad. Por ejemplo: si el primer hueco de la plantilla abstractas es un museo, este nuevo recomendador se encargará de definir exactamente a qué museo se irá en función de las preferencias del usuario.

  - ***Paso 7:*** se devuelve la plantilla de actividades ya rellenas como plan propuesto.

[1] El valor de satisfacción varía entre 0 y 1.

C.15. Generación de recomendación grupal (servidor)

- **Precondición**: el cliente Android debe haber solicitado una recomendación grupal al servidor y enviado los parámetros - restricciones - de la misma.
- **Postcondición**: el servidor devolverá al cliente el plan de ocio que mejor encaje con los parámetros de recomendación.
- **Resumen**: este caso de uso corresponde a la continuación de la solicitud de una recomendación por parte del cliente. El principal objetivo es mostrar, de forma muy básica, el funcionamiento de la recomendación en la parte de *backend*. En un capítulo posterior se detallará el algoritmo de recomendación completo.
- **Flujo Normal**: (Figura C.15)
  - ***Paso 1:*** el servidor recibe los parámetros de recomendación del cliente (horario, lugar, actividades, etc.).
  - ***Paso 2:*** se realiza un primer filtrado sobre el número y tipo de actividades a mostrar. Se combinan para ello la predisposición de los usuarios del grupo a realizar planes fuera del horario señalado y sus preferencias por cada tipo de actividad. Por ejemplo: si el usuario solicitante da un margen de 2 horas para un plan, marca los 5 tipos de actividad para ser realizados en ese intervalo y, además, en media, a los integrantes del grupo no les gusta que se les recomienden planes fuera del horario previsto; entonces el sistema recortará las 5 actividades solicitadas y dejará solo una de ellas - la preferida del grupo -. Por otro lado, si otro usuario solicitante marca un intervalo de 2 horas para el plan con 5 actividades pero al grupo, en media, no le importa que el plan propuesto se salga del horario, Madrid Live no recortaría ninguna actividad y alargaría el plazo del plan previsto.
  - ***Paso 3:*** a continuación se llama al recomendador de plantillas concretas para buscar planes realizados por otros usuarios. Para cada plan recuperado se comprueba si cumple las restricciones impuestas (horario, lugar, etc.). En caso afirmativo, se calcula un valor de "satisfacción", Éste indicará el nivel de agrado de dicho plan para el grupo de usuarios.
  - ***Paso 4:*** si se encuentra algún plan entre las plantillas concretas con valor de satisfacción mayor a 0.75[2] entonces se devuelve dicho plan como propuesta para el grupo.

[2]El valor de satisfacción varía entre 0 y 1.

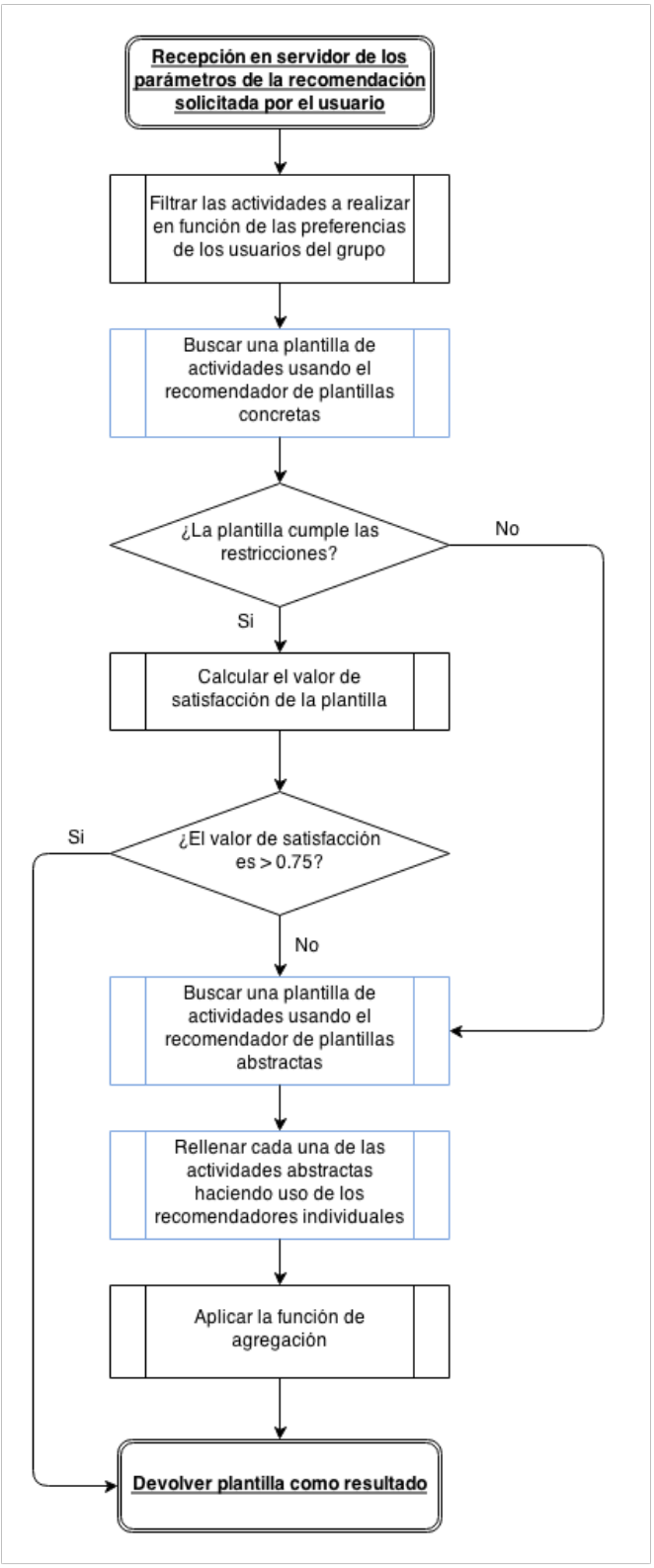


Figura C.15: Diagrama de flujo: generación de una recomendación grupal

- ***Paso 5:*** si no se ha encontrado un plan dentro de las plantillas concretas, entonces se llamará al recomendador de plantillas abstractas. Éste se encarga de generar un grupo de actividades no definidas. Por ejemplo: un paseo y un parque de 16:00 a 20:00.
- ***Paso 6:*** acto seguido, para cada actividad abstracta presente en la plantilla anterior, se llamará al correspondiente recomendador de actividad. Por ejemplo: si el primer hueco de la plantilla abstractas es un museo, este nuevo recomendador se encargará de definir exactamente a qué museo se irá en función de las preferencias del usuario. La principal diferencia frente a la recomendación individual es que, en este caso, se solicita una actividad concreta para cada usuario del grupo y luego se selecciona la más apropiada para todos los integrantes aplicando una función de agregación.
- ***Paso 7:*** se devuelve la plantilla de actividades ya rellenas como plan propuesto.

Bibliografía

Bridge, D., Göker, M. H., McGinty, L. y Smyth, B. *Case-based recommender systems*, vol. 20:3. 2006a.

Bridge, D., Göker, M. H., McGinty, L. y Smyth, B. *Case-based recommender systems*, vol. 20:3. 2006b.

Gedikli, F. Recommender systems and the social web. leveraging tagging data for recommender systems. 2013.

Gilbert, K., E. y Karahalios. Predicting tie strength with social media. páginas 211–220, 2009.

Golbeck, J. Combining provenance with trust in social networks for semantic web content. páginas 101–108, 2006a.

Golbeck, J. Generating predictive movie recommendations from trust in social networks. páginas 93–104, 2006b.

Lara Quijano Sánchez, B. D. A., Juan A. Recio García. Personality and social trust in group recommendations. 2010.

Masthoff, J. *Group recommender systems: Combining individual models.*. 2011.